



**Calhoun: The NPS Institutional Archive**  
**DSpace Repository**

---

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

---

1994-03

# Improving detection and acquisition in Janus(A) using the Pegasus database

Dau, Frederick W.

Monterey, California. Naval Postgraduate School

---

<http://hdl.handle.net/10945/30892>

---

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

*Downloaded from NPS Archive: Calhoun*



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

**Dudley Knox Library / Naval Postgraduate School**  
**411 Dyer Road / 1 University Circle**  
**Monterey, California USA 93943**

<http://www.nps.edu/library>

**NAVAL POSTGRADUATE SCHOOL**  
**Monterey, California**



**THESIS**

IMPROVING DETECTION AND ACQUISITION  
IN JANUS(A) USING THE PEGASUS DATABASE

by

Frederick W. Dau IV

March, 1994

Thesis Advisor:

Morris Driels

Approved for public release; distribution is unlimited.

Thesis  
D16185



DUDLEY KNOX LIBRARY  
NAVAL POSTGRADUATE SCHOOL  
MONTEREY CA 93943-5101

# REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 11 March 1994.	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE IMPROVING DETECTION AND ACQUISITION IN JANUS(A) USING THE PEGASUS DATABASE		5. FUNDING NUMBERS	
6. AUTHOR(S) Frederick W. Dau IV			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) Janus(A) is a wargaming simulation in which opposing army forces interact within a pre-specified terrain database. The smallest unit of area in the terrain is 100 meters by 100 meters. Work is described which utilizes a one meter by one meter terrain database, allowing existing detection and acquisition algorithms to become more dynamic and more realistic. Development of a line of sight algorithm used to incorporate the higher resolution database is discussed. The resulting algorithm is demonstrated in a simulation of a target moving through obstructions contained within a limited terrain described by the one meter database.			
14. SUBJECT TERMS Target Detection, terrain database, target acquisition, line of sight, Janus, Pegasus.		15. NUMBER OF PAGES 184	
		16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)

Prescribed by ANSI Std. Z39-18

Approved for public release; distribution is unlimited.

IMPROVING DETECTION AND ACQUISITION IN  
JANUS(A) USING THE PEGASUS DATABASE

by

Frederick W. Dau IV  
Lieutenant, United States Navy  
B.S., United States Naval Academy, 1987

Submitted in partial fulfillment  
of the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

March 1994

Author:

Frederick W. Dau IV

Approved by:

Morris Driels, Thesis Advisor

Matthew D. Kelleher, Chairman  
Department of Mechanical Engineering

## ABSTRACT

Janus(A) is a wargaming simulation in which opposing army forces interact within a pre-specified terrain database. The smallest unit of area in the terrain is 100 meters by 100 meters. Work is described which utilizes a one meter by one meter terrain database, allowing existing detection and acquisition algorithms to become more dynamic and more realistic. Development of a line of sight algorithm used to incorporate the higher resolution database is discussed. The resulting algorithm is demonstrated in a simulation of a target moving through obstructions contained within a limited terrain described by the one meter database.

THS/12  
D16/185  
C.2

## TABLE OF CONTENTS

I.	BACKGROUND . . . . .	1
A.	INTRODUCTION . . . . .	1
B.	THE JANUS MODEL . . . . .	2
C.	TARGET ACQUISITION IN JANUS . . . . .	3
1.	Background . . . . .	3
2.	Signature Attenuation . . . . .	4
3.	Resolvable Cycles . . . . .	5
4.	Acquisition . . . . .	6
D.	LINE OF SIGHT IN JANUS . . . . .	8
1.	DOLOS Subroutine . . . . .	8
2.	Celski's Claim . . . . .	10
a.	Database Representation . . . . .	10
b.	PLOS Calculation . . . . .	11
E.	THE PEGASUS DATABASE . . . . .	13
F.	OBJECTIVE . . . . .	16
II.	MODELING . . . . .	18
A.	TERRAIN MODELING . . . . .	18
B.	DATA CONVERSION . . . . .	19
C.	TEST BATTLEFIELD CONSTRUCTION . . . . .	22
III.	LINE OF SIGHT/ACQUISITION . . . . .	25

A.	DEVELOPMENT . . . . .	25
B.	DETERMINING THE NUMBER OF LINE'S OF SIGHT . . . . .	25
C.	STEPPING THE LINE OF SIGHT . . . . .	29
D.	GRID EVALUATION . . . . .	32
	1. Line of Sight Height . . . . .	32
	2. Terrain Intersection . . . . .	32
	3. Undercover Index . . . . .	33
	4. Terrain Type . . . . .	34
E.	ATTENUATION OF THE LINE OF SIGHT . . . . .	35
	1. Tree Density . . . . .	35
	2. Distance of Foliage from Target . . . . .	36
	3. Multiple Intersections . . . . .	38
F.	LINE OF SIGHT REACHES TARGET . . . . .	39
G.	MODIFYING THE LINE OF SIGHT . . . . .	39
	1. Incident Angle . . . . .	39
	2. Atmospheric Attenuation . . . . .	42
	3. Applying Modification Factors . . . . .	42
IV.	SIMULATION . . . . .	44
A.	DETECTION SIMULATION IN THE ONE METER DATABASE . . . . .	44
B.	EXAMINATION OF A SPECIFIC TARGET . . . . .	47
C.	DISCUSSION . . . . .	54
V.	DISCUSSION . . . . .	56
A.	EFFECTS OF ONEMETER ON ACQUISITION . . . . .	56
	1. Replacement of DOLOS . . . . .	56

B.	EFFECT OF ONEMETER ON OTHER SUBROUTINES . . . .	57
1.	Direct Effects . . . . .	57
2.	Indirect Effects . . . . .	61
C.	IMPLEMENTATION INTO JANUS . . . . .	62
1.	Database Manipulation . . . . .	62
2.	Target Representation in Janus . . . . .	63
3.	Three Dimensional Target Surfaces . . . . .	63
D.	VERIFICATION . . . . .	64
1.	Program Compatibility with Janus . . . . .	64
2.	Reality Check . . . . .	65
VI.	CONCLUSIONS AND RECOMMENDATIONS . . . . .	67
A.	CONCLUSIONS . . . . .	67
B.	RECOMMENDATIONS . . . . .	67
	LIST OF REFERENCES . . . . .	69
	APPENDIX A - A.D. KELLNER'S MEMORANDUM FOR THE RECORD .	70
	APPENDIX B - DOLOS . . . . .	81
	APPENDIX C - HITREE. . . . .	86
	APPENDIX D - A STUDY OF THE LINE OF SIGHT CALCULATIONS AND DATA BASE FOR THE JANUS(A) MODEL . . . . .	88
	APPENDIX E - CONVERT . . . . .	101
	APPENDIX F - BATFIELD . . . . .	103
	APPENDIX G - ONEMETER . . . . .	106
	APPENDIX H - DETECT . . . . .	119
	APPENDIX I - HANDOFF . . . . .	133

APPENDIX J - TARGET ACQUISITION IN JANUS ARMY . . . . .	139
APPENDIX K - EXCERPT FROM: JANUS(T) DOCUMENTATION AND USERS MANUAL . . . . .	155
APPENDIX L - DESCRIPTION OF BIBLIOGRAPHY ITEM CONTENTS	169
BIBLIOGRAPHY . . . . .	172
INITIAL DISTRIBUTION LIST . . . . .	173



# LIST OF TABLES

Table I	RELATIONSHIP OF DENSITY LEVEL TO TREE HEIGHT AND PLOS IN JANUS (A) . . . . .	9
Table II	TPLOS FOR A TARGET IN A GRID OF GIVEN DENSITY	12
Table III	SIMULATION DATA . . . . .	46
Table IV	SUBROUTINES THAT CALL DOLOS . . . . .	59

## LIST OF FIGURES

Figure 1: Definition of Pegasus Database Structure . .	14
Figure 2: Database Element Definition . . . . .	15
Figure 3: Available Terrain Obstacle . . . . .	20
Figure 4: Battlefield Sample . . . . .	24
Figure 5: Line of Sight Evaluation . . . . .	26
Figure 6: Number of LOS Dependence on Aspect . . . . .	27
Figure 7: Target Model . . . . .	28
Figure 8: LOS Path . . . . .	29
Figure 9: LOS Dependence on Obstacle Location . . . . .	31
Figure 10: LOS Ground Intersection . . . . .	32
Figure 11: Unimpeded LOS . . . . .	33
Figure 12: Passage of LOS Below Undercover . . . . .	34
Figure 13: Views when One Observer is Concealed in an Obstacle . . . . .	37
Figure 14: Attenuation as a Function of Distance from Sensor . . . . .	38
Figure 15: Projected Area of a Surface Parallel to the Y- Axis . . . . .	40
Figure 16: Projected Area of a Surface Parallel to X- Axis . . . . .	41
Figure 17: Simulation of Target Detection . . . . .	45
Figure 18: Target Faces . . . . .	47
Figure 19: LOS Path to Face A. . . . .	49

Figure 20: LOS Path to Face B. . . . .	49
Figure 21: LOS Path to Face C. . . . .	50
Figure 22: LOS Path to Face D. . . . .	50
Figure 23: LOS Path to Face E. . . . .	52
Figure 24: LOS Path to Face F. . . . .	52
Figure 25: LOS Path to Face G. . . . .	53
Figure 26: LOS Path to Face H. . . . .	53

## **I. BACKGROUND**

### **A. INTRODUCTION**

As the nation trims down both the size and budget of the military effective use of funds is becoming increasingly more important. Finding ways to reduce costs without sacrificing readiness is paramount. One manner to reduce costs yet retain performance is through simulation. Manipulation of forces across a computer screen is much more cost effective, and safer, than mobilization of large forces for training exercises. Simulation is not a new concept, but one whose usefulness is becoming more prominent.

The Army has had a wargaming simulation known as Janus(A) for over a decade. Janus(A) uses a database defined by large 10000 square meter terrain grids to create battle simulations for use in planning and evaluation of various combat scenarios. Recently a much higher resolution database known as Pegasus has been developed using terrain grids of only one square meter. If the new Pegasus database could be incorporated into the Janus structure, then it would be theoretically possible to achieve a simulation which more closely approaches reality.

## **B. THE JANUS MODEL**

The Janus simulation was originally fielded in 1978. The simulation was named after Janus, the two faced Roman god of portals who guarded the gates of Rome by looking in two directions at the same time. Originally intended for use as a nuclear effects model, Janus became a useful training tool as well as a model for combat development research. Janus was updated to version 2.1 in January 1992. [Ref. 1:p. 1]

The Janus model simulates battle between Blue and Red units. Forces from each side up to brigade and regimental sized units are controlled by two or more users operating from separate computer terminals. [Ref. 1:p. 1]

Each terminal presents the operator with a high resolution graphical depiction of friendly forces, detected enemy forces, terrain contours, and a variety of other useful information. The Janus model is composed of 13 executable FORTRAN programs. These programs are divided into three major groups: those used to create and maintain the database; those which build verify and run the scenario; and those used to analyze scenario results [Ref. 1:p. 5].

The database used by Janus is a digitized terrain developed by the Defense Mapping Agency. The terrain is divided into grids 100 meters on a side, each of which is numerically represented by values characteristic of the particular grid. [Ref. 2:p. 3]

### **C. TARGET ACQUISITION IN JANUS**

In order to develop a program which makes use of a new higher resolution database it is first necessary to gain a basic understanding of how acquisition is currently determined. The next four sections are a summarization of A.D. Kellner's 14 July, 1992 memorandum for the record in which he discusses detection in Janus. This memorandum as originally written is found in Appendix A.

#### **1. Background**

Acquisition in the Janus model is based on the mathematical detection model developed by the Night Vision and Electro-Optics Laboratory (NVEOL). This model is based on a concept involving the computation, for a specific sensor-target pair, of the number of resolved cycles across a target's critical dimension.

Imagine a pattern of stripes or bars equal in length and alternating in color at the target's location. Let the contrast between the colors of the pattern be the same as the contrast between the image of the target and its background. The length of the pattern is the same as the target's minimum presented dimension. Slowly decrease the width of the stripes until the minimum width at which the observer can still distinguish the individual stripes is reached. The number of pairs now contained in the minimum presented area is called

the number of resolvable cycles for that target-sensor combination.

The concept of resolvable cycles is very useful in the computation of detection probabilities. The NVEOL model defines two probabilities associated with the detection of a given target by a given sensor. First is the probability that the target will be detected by the sensor given infinite time, PD. This quantity is also known as p-infinity. Second is the probability that the target will be detected during the time it is within the sensor's field of view, given that it can eventually be detected. This is called  $P(t)$ . Both quantities of PD and  $P(t)$  are functions of the number of resolvable cycles,  $N$ , which can be resolved by a given sensor-target pair. The probability that a sensor can discriminate the target once detected is also a function of  $N$ .

The portion of the NVEOL model relevant to Janus consist of the three following steps: 1) calculate the attenuation of the target's signature along the line of sight 2) given the target's signature at the sensor, calculate the number of cycles the sensor can resolve across the target's critical dimension 3) Given  $N$ , determine if the target can be detected, acquired, and recognized.

## **2. Signature Attenuation**

Signature attenuation between the target and the sensor is caused by atmospheric effects as well as objects

obscuring path of the line of sight which will be called large area smoke. Let:

$St$  = signature at target

$Ss$  = signature at sensor

$T1$  = transmission of normal atmosphere

$T2$  = transmission of large area smoke

and

$$Ss = St * T1 * T2 \quad (1)$$

For optical sensors,  $St$  is the optical contrast of the target which is a part of the master database, and thus remains the same for a target during a Janus run.

### 3. Resolvable Cycles

The performance of a sensor is represented by a curve of resolvable cycles per milliradian (CMR) as a function of the target signature measured at the sensor. This can be expressed mathematically as follows:

$$CMR = f(Ss) \quad (2)$$

However there is no analytical equation to describe this function, so this function is input in the master database for each sensor as a tables of values. Entering the table of the specific sensor-target pair with the signature at the sensor,  $Ss$ , produces an output of CMR.



Once the number of resolvable cycles is obtained the number of cycles actually resolved by the sensor is obtained by:

$$N = CMR * \frac{TDIM}{Range} \quad (3)$$

where

TDIM = target's minimum presented dimension in meters

Range = sensor to target range in kilometers.

TDIM is obtained from the master database for the particular target and range is provided by the simulation.

#### 4. Acquisition

After the number of cycles resolved on the target is known the probability of detection and time till detection,  $P(t)$ , may be determined. The probability of detection is given by

$$PD = \frac{CR^W}{1 + CR^W} \quad (4)$$

where

$$W = 2.7 + 0.7^{CR}$$

$$CR = N/N50$$

$N$  = number of resolvable cycles

$N50$  = median number of resolvable cycles required for eventual detection

Note that when  $N = N50$  the equation 4 goes to 0.5, meaning that for a random sample of observers, half will require less

than N50 resolvable cycles to detect the target and half will require more. Janus uses the following N50 criteria for detection and subsequent target discrimination:

- 1.0 cycles Detection: sensing a foreign object is present
- 2.0 cycles Aimpoint: ability to aim at a target
- 3.5 cycles Recognition: ability to distinguish target class, such as tank, truck, jeep
- 6.4 cycles Identification: ability to classify the target as specific member of a class

At the beginning of a Janus run each sensor-target pair is assigned a random value which represents the ability of the sensor to detect the corresponding target. This random number is the number of resolved cycles required to detect and is called the detection threshold. When N is greater than or equal to the detection threshold it is said that the p-infinity test has been passed and that the target may eventually be detected by that particular sensor. Once the p-infinity test is passed a value for  $P(t)$  is determined using the values of CR and PD. This value is compared to another randomly drawn number, and if it exceeds this random number then detection or another level of discrimination is achieved.

#### D. LINE OF SIGHT IN JANUS

##### 1. DOLOS Subroutine

If sufficient obstructions exist between the sensor and target so as to prevent the target from being seen then acquisition cannot occur. Therefore before acquisition is considered the existence of an acceptable line of sight (LOS) between the sensor and target must be verified. The DOLOS subroutine called by Janus determines the probability of line of sight (PLOS) existing between a sensor and target based on their respective locations and elevations. A b r i e f description of the DOLOS subroutine found in Appendix B follows.

When called DOLOS first retrieves sensor and target grid locations and elevations. Based on the target location a temporary probability of line of sight (TPLOS) is also initialized.

Each grid within the Janus database is assigned a density value. In order to select the density value the area is surveyed for the number and height of vegetation. Based on these findings the grid is assigned its density. Eight different density values exist and each is assigned a tree height and a grid PLOS as shown in Table 1 [Ref. 4:p. 4]. Once this density value is assigned the entire grid will have the same height and PLOS.

The function HITREE, found in Appendix C, is called to obtain the grid height and density value. The target is assumed to make the best use of terrain obstacles in its grid for concealment, thus the PLOS value for the grid is squared and made the initial value of TPLOS.

Next, given the coordinates of the sensor and target, the slope of the line between the two points is determined. Based on the slope DOLOS incrementally steps one grid at a

**Table I** RELATIONSHIP OF DENSITY LEVEL TO TREE HEIGHT AND PLOS IN JANUS (A)

DENSITY LEVEL	TREE HEIGHT (meters)	PLOS
0	0	1
1	3	2
2	7	4
3	9	6
4	10	7
5	11	8
6	13	9
7	14	10

time from the sensor to the target in the horizontal plane. The slope in the vertical plane is also calculated, and the incremental height change is added to the sensor height each step. In each grid along the determined path of the LOS HITREE retrieves grid tree height and density factor. The tree height is added to the grid's ground elevation and then

compared to the elevation of the LOS in that grid. If the LOS height is less than that of the grid's base elevation then the LOS calculation is terminated because the ground is in the way, and no LOS exists. If the LOS height is greater than the total elevation of the grid cell then the LOS is unimpeded and the program moves on to the next grid cell. If the LOS height falls between the base grid elevation and the total elevation including obstacles then value of TPLOS is multiplied by the value of PL retrieved from HITREE. This is continued for each grid until the target grid is reached or until TPLOS falls below a value of 0.01. If TPLOS falls below 0.01 then the LOS has been completely attenuated by the terrain features along its path and does not exist.

When DOLOS completes its run it passes the final value of TPLOS back to the calling routine as PLOS.

## **2. Celski's Claim**

In September of 1992 Major Robert Celski conducted a study of the LOS calculations and data base for the Janus model in which he reported several problems. This report is contained in Appendix D.

### **a. Database Representation**

Celski's first point is that the vegetation and urban terrain heights and densities may not be properly represented in the Janus model or database. Celski correctly pointed out that the assignment of tree heights to a given

grid are poorly distributed. Tree height assignment is weighted towards taller trees. This leaves trees with heights between zero and six meters to be classified as either zero or three meters tall. Meanwhile trees seven meters or higher have six different height classifications. Refer again to Table 1. This leads to an unrealistic portrayal of the vegetation in a grid. He also pointed out that the assignment values for a probability of LOS (PLOS) at a given density are backwards, i.e., a low density grid has a small PLOS while a high density grid is given a high PLOS. Celski believed this was a problem, however this is only a matter of aesthetics. It is the manner in which the PLOS value is used that is important, not the numerical value. This leads to Celski's second point.

***b. PLOS Calculation***

The use of PLOS in the DOLOS subroutine and the HITREE function is faulted in such a way that a LOS can exist only if the target is located in a grid which has no vegetation. In the function HITREE the PLOS value retrieved is multiplied by 0.01 before being returned to DOLOS. In establishing an initial TPLOS for the target grid when the value PL is returned from HITREE it is multiplied by itself and is called TPLOS. As shown in Table 2 the initial value of TPLOS is already at or below the minimum allowed value of 0.01. [Ref. 4:p. 8]

**Table II** TPLOS FOR A TARGET IN A GRID OF GIVEN DENSITY

DENSITY LEVEL	TPLOS
1	0.0004
2	0.0016
3	0.0036
4	0.0049
5	0.0064
6	0.0081
7	0.010

There are two problems here. First, the one case which will allow a LOS is for a target concealed in the most dense grid, which clearly makes no sense. Looking again at Table 1, with increasing grid density the reduction of TPLOS becomes less and less, which is clearly contrary to common sense. As the grid density increases the PLOS of a grid should become smaller, not larger. Second, it makes sense there must be cases for non-zero density level grids that a LOS exists. These errors point out the incorrect use of density factors in the Janus model.

While Celski was on the correct path and did point out a problem he went one step too far in his calculations. He claimed that the value returned from HITREE was multiplied by itself twice, essentially cubing the original value. In actuality the value is only squared.

Celski made an error when considering modification of TPLOS later in DOLOS. He stated that TPLOS was multiplied by the initial target grid HITREE value when in fact it is an interfering grids value of PLOS that makes the modification, and not the PLOS value from the target grid.

#### **E. THE PEGASUS DATABASE**

The database known as Pegasus is actually the Perspective View Database (PVDB) which was originally created in support testing focal plane seeker guided weapons. The PVDB is a geographic database created from over 200 aerial photographs which contains information required for perspective view generation. The database covers a rectangular area of Fort Hunter Liggett covering more than 400 square kilometers. The database comes in resolutions of 1,4,16, and 64 meters. The one meter post will be considered. [Ref. 5:p. 1]

Each post is described by a 32 bit number which contains a wide array of information about the post. The structure of this number is described by Figure 1. As a binary number, each place represented in Figure 1 contains either a one or a zero.

Visualization of the components making up each post is made easier by examining Figure 2 [Ref. 5:p. 3]. Figures 1 and 2 show that the Pegasus database is essentially composed of a number of 32 bit binary numbers, each of which is made up of nine different elements. The grid elevation may be defined



3 1 0 9 8 7 6 5 4 3 2 1											2 0	9 8 7 6 5 4				1 3 2 1 0	9 8 7 6				5 4 3 2 1 0			
ELE											E L 2	UCI	NOR				VGH	VID	N S A S T B	GSV				

Element Code	Number of bits	Maximum Value	Description
ELE	11	2047	Elevation in meters
EL2	12	4095	Elevation in half meters
UCI	2	3	Under Cover Index
NOR	4	15	Surface Normal Indicator
VGH	4	15	Vegetation Height Index
VID	2	3	Vegetation ID
NAT	1	1	Nature Bit
SSB	1	1	Sun Shade Bit
GSV	6	63	Gray Shade Value

**Figure 1:** Definition of Pegasus Database Structure

to the nearest half meter using the ELE and EL2 elements. The NAT bit defines whether or not the feature is man made or natural, and the UCI, NOR, VGH, and VID bits are combined to describe a variety of different sizes and shapes of objects. A wide variety of colors may be represented using the GSV bit, and the database can even be related to the passage of the sun using the SSB bit. In all, the Pegasus database has the potential to describe a selected terrain in great detail and complexity. [Ref. 7]

This 32 bit binary number is stored as a decimal and must be manipulated to extract the desired information. The FORTRAN language contains a function which makes this

bit assignments      3      2      1      0

10987654321098765432109876543210	ucd	nor	vgh	ucd	nor	gv
----------------------------------	-----	-----	-----	-----	-----	----

Where:

ele: elevation above sea level in 5 meters

ucd: under cover index

nor: surface normal

vgh: vegetation height

vid: vegetation ID

n: nature or manmade

s: sun/shade bit

gv: grayshade

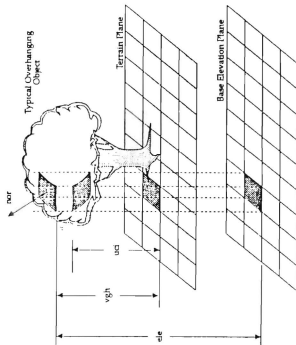


Figure 2. Database Element Definition

seemingly difficult task simple, IBITS. The IBITS function is defined and operates as follows:

$$X=IBITS(A,J,K)$$

extracts K bits from the variable A beginning with the Jth bit. Thus if it is desired to find the value for UCI of a number B, then referring to Figure 1 and the bit locations, the proper command is IBITS(B,18,2). IBITS will then convert the number B into a binary number, extract the first two bits beginning with bit 18, and present the result as the decimal representation of the two bit number. [Ref. 6:p. D.42]

#### F. OBJECTIVE

The task at hand is to develop a new code for Janus which improves the method by which target detection and acquisition are determined. The new code should be designed with several goals in mind.

- 1) It must be able to read the new Pegasus 1 meter database and be able to extract the information contained within.

- 2) It should be able to take advantage of the high level of detailed information provided by Pegasus, and use this information to determine whether or not a line of sight exists, and if so, determine the quality of that line of sight.

- 3) The new algorithm should be able to interact with the current Janus coding and be able to make use of existing Janus

subroutines when appropriate. It must be able to return useful information to the Janus program in a form that can be used.

If these goals can be met then it will be possible to obtain a more accurate and realistic line of sight and acquisition algorithm for use in the Janus program.

## II. MODELING

### A. TERRAIN MODELING

In order to develop an acquisition algorithm which uses a new and more defined database it is necessary to have an arena in which it can be tested. At first it would seem that the best test ground would be an actual extract from the terrain of Fort Hunter Liggett itself. However, this is not practical for two reasons. First, it would be difficult to use Fort Hunter Liggett terrain as a standard test case because it comes unknown. The exact make-up of the terrain would be initially unknown, and decoding it into a known would take a great deal of time. Secondly, not all of the Fort Hunter Liggett exercise areas have been surveyed for trees. In fact the original tile provided contained no features at all except ground elevation.

Development of a known terrain is therefore desirable. Testing a new algorithm against a known and well defined terrain allows for a more critical examination of the algorithm, as well as making trouble-shooting easier.

Obstacles which may impeded the line of sight must be such that they can be defined by the Pegasus database. The Pegasus database has the capability to define objects with a greater deal of complexity. The object may not only have a specific

height and width, but it also may have its own particular structural make up. An object can be defined as a tree or a building, it may be partially transparent or completely opaque, it may affect the line of sight from the ground up or may only take effect above a certain height. A sample terrain used for testing and evaluation must incorporate each these special capabilities of the database.

A variety of different obstacles have been selected for placement within the database in a manner which may be described within the Pegasus database structure. These objects are three dimensional in nature and are shown in Figure 3. As shown, there are four sizes of trees and two possible building types available for placement in the test area. Each block in the figure represents a one meter cube.

## **B. DATA CONVERSION**

As mentioned above, in order to create a useful test field it is first necessary to build a data base which has the same format as the Pegasus database. As discussed in section II.1.D, the Pegasus database is made up of integer numbers corresponding to a 32 bit number. This 32 bit number also contains nine separate field descriptions. The objective then is to create a routine which takes each inputted data set, combines them properly, and then converts the resultant binary number into an equivalent integer. The program called CONVERT in Appendix E meets these objectives.

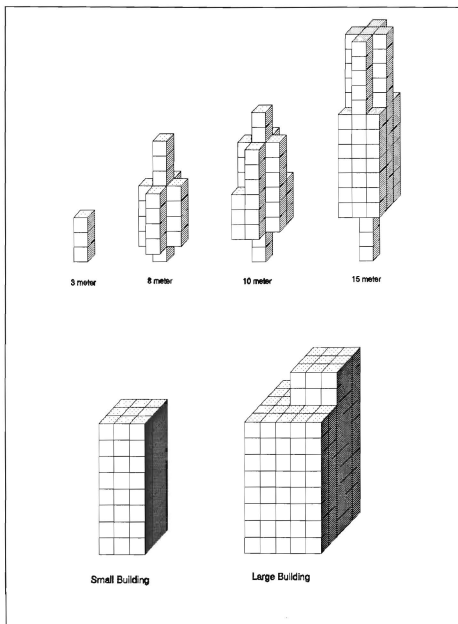


Figure 3: Available Terrain Obstacle

CONVERT first loops through all the data fields and converts the integer entered into a binary number. The operator is prompted to enter the number required for the particular field and is given a range of acceptable values. At the same time the position of the lowest bit within the 32 bit number is recorded. Once this number is read the conversion to binary begins, and is perhaps best illustrated by walking through the conversion of an actual number.

Consider the number nine entered for the vegetation height index (VGH). From Figure 1 it is seen that the lowest bit of the field is the tenth bit in the overall 32 bit number. First the number is checked to see if it is zero, in which case it is already in binary form, nine is not zero so the program continues. In order to find the power of two,  $n$ , required for a non-zero number the log of the entered number is taken and then divided by the log of two.

$$n = \frac{\log(9)}{\log(2)} = 3.169925 \quad (5)$$

Next the number  $n$  is converted into an integer,  $m$ . The low bit value for the data field plus  $m$  becomes one. Two is raised to the  $m^{\text{th}}$  power and the result is subtracted from the original number.

$$9 - 2^3 = 1 \quad (6)$$

If the remainder is zero the conversion to binary is complete, if not then the process is repeated using the remainder.



$$n = \frac{\log(1)}{\log(2)} = 0.0 \quad (7)$$

$$1 - 2^0 = 0 \quad (8)$$

Now the remainder is zero and the binary number '1 0 0 1' occupies the thirteenth through tenth places respectively.

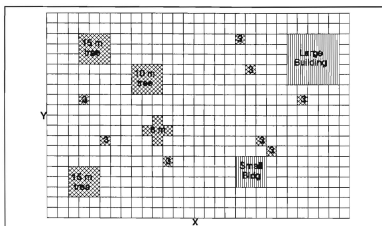
After converting all inputted values to binary and establishing their positions in the 32 bit binary number, the number must be converted to a decimal integer. A binary number is composed of ones and zeros. Thus conversion to an integer is simple. Starting with the number zero and counting to the left, each place represents two raised to its corresponding place number. That number is multiplied by either one or zero, whichever is present in the binary number, and the value obtained from each calculation is summed. The final sum is the integer value of the binary number. The resulting binary number can now be used to describe the one meter volume of terrain, and combination of several of these pieces will describe an object.

### C. TEST BATTLEFIELD CONSTRUCTION

Assembly of the pieces of data created using CONVERT and organization of the data into a useful format is the purpose of the program BATFIELD, found in Appendix F. BATFIELD is an interactive routine that places the objects shown in Figure 3 on a 250 meter by 250 meter playing field.

Using the program CONVERT each piece of the Figure 3 obstacles is converted into its proper Pegasus format. This number is assigned a variable value in BATFIELD. Initially all terrain grids are initialized to have zero elevation and be covered by grass. Then BATFIELD begins a loop requesting the location of a particular obstacle. When the user enters the desired coordinates of the center of the object BATFIELD recalls the appropriate data variables and assigns them to the proper locations within the test area. When all objects have been entered BATFIELD transfers the values to a file created in a format which may be read by the line of sight calculation program.

The battlefield created contains a number of each of the allowed objects scattered across the area in a manner to allow complete testing of the new line of sight routine. When viewed from above a small section of the field appears as shown in Figure 4. Each grid represents one square meter in area and the obstacles represented are labeled.



**Figure 4:** Battlefield Sample

### III. LINE OF SIGHT/ACQUISITION

#### A. DEVELOPMENT

In order to achieve acquisition, a line of sight (LOS) must first be determined to exist, and the quality of the LOS must be examined. In the program ONEMETER, found in Appendix G, the LOS is obtained in a manner similar to the existing Janus subroutine previously described, DOLOS. However, instead of moving through grids 100 meters on a side the LOS will now move through grids one meter on a side. Given the sensor and target location the algorithm proceeds roughly as shown in the block diagram of Figure 5. Each step will be described in some detail. Comments are provided within ONEMETER to assist in following the program logic.

#### B. DETERMINING THE NUMBER OF LINE'S OF SIGHT

In the older database the target size was essentially insignificant in comparison to the size of the grid, and thus the path between the sensor and target could be represented as a single LOS. However, in the one meter data base a target may extend over several grids thus allowing several possible LOS. Consider situations present in Figure 6. The sensor is considered to originate from a very small point in space, and is assumed to only occupy one grid. In the case of Sensor 2, it is possible to see the vertical faces of blocks 1 and 3 of

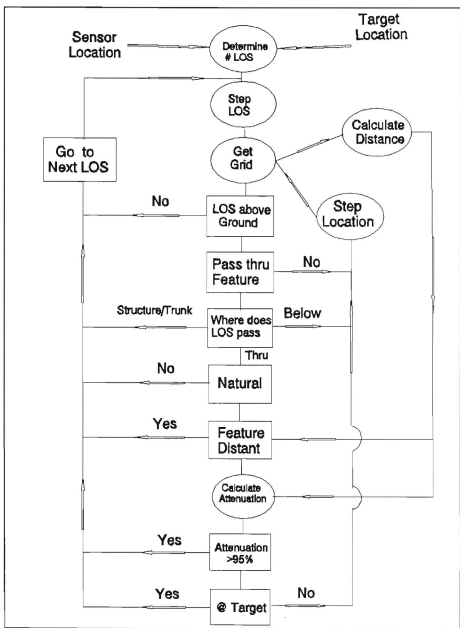
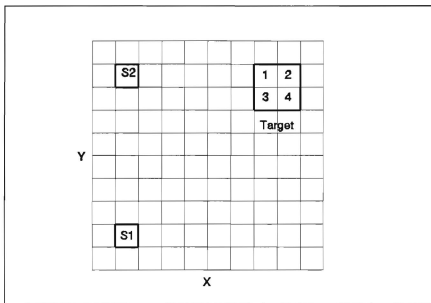


Figure 5: Line of Sight Evaluation

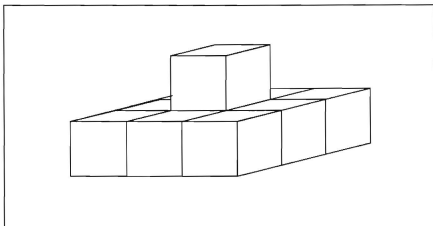


**Figure 6:** Number of LOS Dependence on Aspect

the target. Sensor 1 on the other hand has the possibility of seeing the vertical faces of blocks 1 and 3 as well as the horizontal faces of blocks 3 and 4 of the target. It is obvious from this example that not only can multiple LOS exist, but the number of LOS which may exist will be dependent on the spatial orientation of the sensor with respect to the target.

In order to examine this aspect a theoretical target is generated with a shape compatible with the Pegasus database. For illustration purposes this target will be a square three meters by three meters one meter high, with another one meter

by one meter by one meter block sitting on the center of the three by three. The target depicted is shown in Figure 7.



**Figure 7:** Target Model

This will give a theoretical target which presents itself in three dimensions.

The subroutine ASPECT, included in the program ONEMETER, takes the inputted target location and places the generated target around the central coordinate on the playing field. Then based on the relationship between the target and sensor the number of target faces which present themselves for detection by the sensor are determined. A maximum of eight faces and a minimum of four faces of the target may be visible at any given time. In calling these faces "visible" it is only meant to say that given a completely unobstructed view of the target this many faces could be seen by the sensor. Once the number of LOS is determined each is evaluated as follows.

### C. STEPPING THE LINE OF SIGHT

Given the grid positions of the target and sensor the difference between the X and Y coordinates is determined, RDX and RDY respectively. This is done in the same manner as with the 100 meter grids, except one meter grids are now used. Whichever difference is larger is chosen to be the direction in which one meter steps are taken. The other direction steps in one meter blocks when the LOS passes over that block, based on the slope calculated between the coordinates. A single block is passed through in each step. Figure 8 provides an illustrative example.

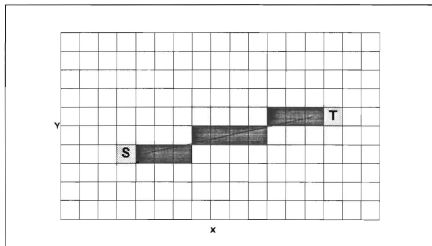


Figure 8: LOS Path

As seen, the difference in X coordinates is larger, so the LOS will step toward the target in the X direction one meter at a time. At each X coordinate the Y value is found using the Y slope and the origination point. The dark line represents the



true path of the LOS between target and sensor, the shaded boxes show how the LOS steps through the grids.

This method of stepping the LOS can result in unique results depending on the distance between the sensor and target, and the location of any intervening obstacle. Consider how the LOS would be stepped from a sensor to a target with three visible faces. Figure 9 shows a top view of the LOS possibilities that may exist. Case 1 shows the LOS path between a sensor and target with no obstacles. The next two cases show how the location of an obstacle can affect the resulting LOS. Note that in Case 2 one branch of the LOS has been blocked, but two of the other LOS branches are unaffected and allow the upper and lower surfaces of the target to be seen. However in Case 3, the obstacle falls closer to the sensor, and blocks the LOS before it branches off, thus no part of the target is seen.

Ideally the line of sight between a sensor and target would encompass a continuous arc across the full target dimension, as shown in Case 4. This anomaly occurs because of the discretization of the problem into finite areas. This problem can only be eliminated by making the areas considered infinitely small so as to create the appearance of an arc such as in Case 4. The computational effort required to achieve this and the difficulty that would be imposed on obtaining a database of such small size make this solution impractical.

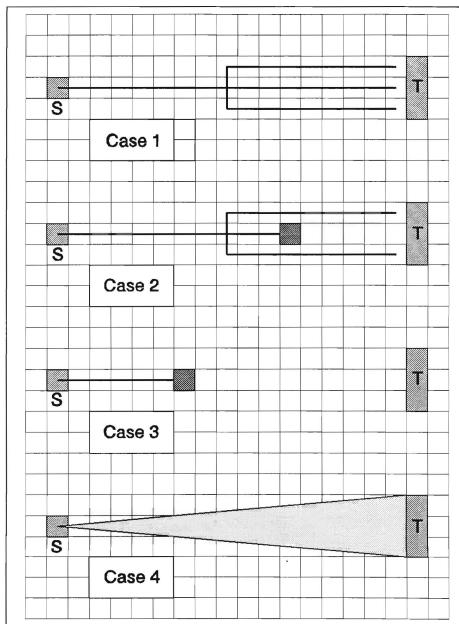
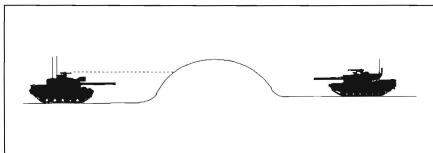


Figure 9: LOS Dependence on Obstacle Location

#### **D. GRID EVALUATION**

##### **1. Line of Sight Height**

When a grid along the LOS is selected the first check done is to determine whether or not the LOS passes above ground level. The height of the LOS is obtained by adding the change in LOS height per grid along the LOS to the previous LOS height. This value is compared to the ground height which is found by subtracting the grid vegetation height from the absolute grid height. If the LOS height is less than the height of the ground, as in Figure 10, then that particular LOS does not exist and the program moves on to the next LOS. Otherwise further analysis of the LOS is conducted.

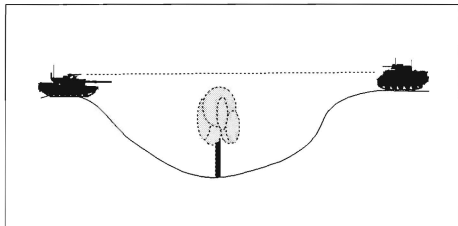


**Figure 10: LOS Ground Intersection**

##### **2. Terrain Intersection**

Next the code determines whether or not the LOS passes through the feature of the grid. The grid feature is any object existing in the grid which has elevation above ground level. The terrain may include both natural as well as

manmade features. The previously calculated LOS height is compared against the absolute grid height. If the LOS height is greater than the absolute grid height, as in Figure 11, the LOS is assumed to pass above the grid terrain unimpeded and the code moves to the next grid in the LOS. If the LOS falls below the grid height then further evaluation of the nature of the terrain is required.



**Figure 11:** Unimpeded LOS

### **3. Undercover Index**

One of the most unique pieces of information contained within the Pegasus database is the undercover index (UCI). The undercover index represents the height of the terrain feature above the ground. This would represent the height of the lowest branch of a tree above the ground or the height of a building overhang above the ground. Knowledge of this

quantity allows for the possibility that the LOS may pass above the ground but below the terrain feature. If the LOS has been determined to pass below the terrain feature height the code compares the LOS height to the undercover index height. If the LOS is below the undercover, as shown in Figure 12, then it is assumed to pass through the grid unimpeded and the code moves to the next grid. Otherwise the LOS intersects the body of the feature.

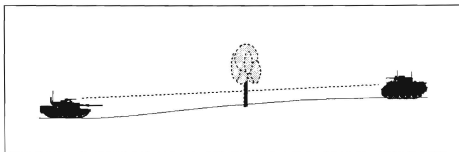


Figure 12: Passage of LOS Below Undercover

#### 4. Terrain Type

The Pegasus database contains a single bit number which indicates whether the terrain feature is natural or manmade. This is called the nature bit. If the LOS intersects the feature, and the feature is manmade, it is assumed that the LOS is completely blocked, and the program moves to the next possible LOS. If the feature is natural then the LOS is attenuated by some factor as discussed in the following section.

## **E. ATTENUATION OF THE LINE OF SIGHT**

The ability to see an object through a tree or shrub is difficult to quantify. This is the first factor which becomes dependent on the perception of the sensor. Clearly this varies from sensor to sensor, where the sensors are actually the human eyes. A trained observer will be better at detection than the casual observer. The level of fatigue and other background distractions will also affect an individuals ability to detect a target, obscured or unobscured.

The branches and leaves of a tree will detract from a LOS passing through the tree. It is also generally true that one is more likely to see through an edge of the tree with few branches than through the center of the tree where a significant amount of foliage may exist. A final observation that can be made is that the further away the tree is the more solid it appears. These three basic principles are applied as follows to determine the final effect on the LOS.

### **1. Tree Density**

How "dense" is a tree? This clearly depends on the type of tree and time of year. Some trees are inherently thicker than others, some lose their leaves while others are evergreens. The Pegasus data base describes vegetation using four different quantities: vegetation identification number (VID), vegetation height index (VGH), nature bit (NAT), and the surface normal (NOR). This allows for many different

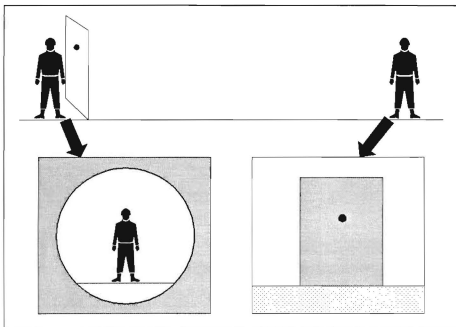
types of vegetation to be designated, each of which may have its own density. Unfortunately the actual manner in which this information is used is not currently available, and an assumption must be made in regard to vegetation density.

Observation of the terrain in the database area indicates that the propensity of vegetation is the live oak. This tree is assumed to have consistent leaf density year round. It will be assumed that passage of the LOS through one meter of vegetation will be attenuated by 30%. This 30% attenuation is based on vegetation one meter from the sensor.

## **2. Distance of Foliage from Target**

Imagine two individuals separated by a wall with a small hole in it. Consider Figure 13. One individual is right next to the wall looking out the hole while the other is some distance away from the wall. The person looking through the hole can see the entire field of view beyond the wall including the other person, while the individual away from the wall sees only a wall with a small hole in it. Using the same principle the effect of foliage distance can be defined.

As the interfering foliage occurs further and further away from the sensor the attenuation must increase until the object appears as a solid object with an attenuation of 100%. Based on purely qualitative observation of several live oaks, this "terminal" distance at which the tree appears solid has been chosen to be 200 meters. The first step taken when the



**Figure 13: Views when One Observer is Concealed in an Obstacle**

LOS has been found to intersect a tree is to check the distance of the tree from the sensor. If this distance is greater than 200 meters the LOS is assumed to be blocked and the program moves to the next LOS. For any trees intersected along the LOS from one to 200 meters away from the target the attenuation factor of the foliage is increased linearly until it has an attenuation factor of 100% at 200 meters.

It is conceivable that the sensor may be concealed within the vegetation. This condition will be represented when the interfering foliage is located one meter or less away from the sensor. In this situation it will be assumed that



the sensor would adjust its vantage point to peer through the foliage, and no degradation of the LOS will be assessed. The graph of Figure 14 shows the relationship between LOS attenuation and foliage distance.

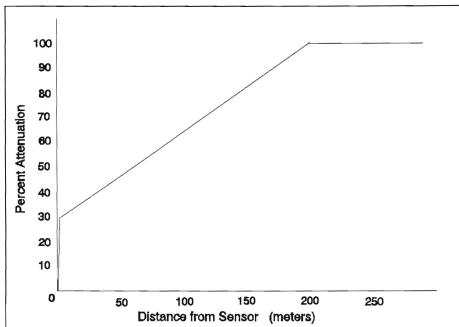


Figure 14: Attenuation as a Function of Distance from Sensor

### 3. Multiple Intersections

Obviously the more vegetation the LOS passes through the more it will be attenuated. During analysis of a single LOS the cumulative attenuation is calculated by adding each incidence of foliage attenuation. Eventually the LOS will be completely lost. This threshold for loss of attenuation is

set at 95%. Once the cumulative attenuation reaches this threshold the LOS is assumed to be blocked and the program moves on to the next LOS.

#### **F. LINE OF SIGHT REACHES TARGET**

If each grid of the LOS passes the above tests and the LOS is not lost due to cumulative attenuation then the LOS being examined between the sensor and target exists.

#### **G. MODIFYING THE LINE OF SIGHT**

After a LOS has been determined to exist it must be modified to account for attenuation due to both foliage and the atmosphere, and for the angle of inclination of the visible surface with the LOS.

##### **1. Incident Angle**

As described in section B of this chapter, the target has several different faces each one square meter in area. If the LOS from the sensor to target is perpendicular to the selected target face then the area presented is one square meter. Any incident angle other than 90 degrees will result in a visible face area of less than one square meter. In order to determine the projected surface area the incident angle of the LOS upon the target face must be determined.

Determination of this angle begins in the ASPECT subroutine where the target is defined. The array 'target' not only contains the X and Y grid locations, but also

information on the surface orientation. In this program the target is symmetrically shaped and has no specific front, back or sides. The target is also allowed to be positioned such that its axis correspond with the main grid axis. Using these restrictions the target can be assumed to move through the grid with only translational motion and no rotational motion, thus maintaining each side facing the same way at all times. Given these assumptions the target information contains data to indicate whether as side is oriented parallel to the main X or Y axis of the battlefield grid. During execution of the ASPECT subroutine this information is passed along to the main program if the side has been determined to possibly be visible.

Once the LOS is found to exist the face orientation is examined. A value of 1 indicates a surface parallel to the Y-axis, a value of 0 indicates it is aligned with the X-axis.

The determination of the incident angle on the target is a simple trigonometric problem as shown in Figure 15.

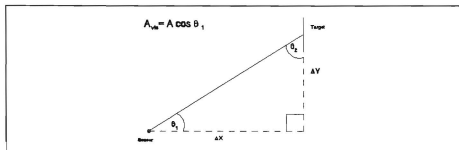
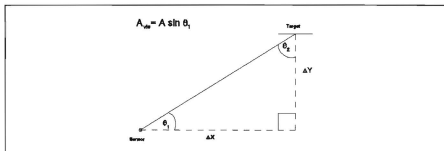


Figure 15: Projected Area of a Surface Parallel to the Y-Axis

The angle  $\theta_1$  is always determined between a line parallel to the X-axis and the LOS. From trigonometry it can be seen that as  $\theta_1$  approaches zero  $\cos\theta_1$  approaches 1. As the angle of incidence upon the target face approaches 90, and the full face area is projected. Conversely, as  $\theta_1$  nears 90, the cosine of the angle nears zero, and the projected area of the face approaches zero. Thus the projected area of a vertical face is the face area times the cosine of  $\theta_1$ , the angle of the LOS with the X-axis. From trigonometry, the cosine of  $\theta_1$  is the opposite side over the hypotenuse of the triangle. Each of these lengths is easily calculated since the grid locations of both the target and sensor are known. Using the same rules of trigonometry it is found that the projected area of a face aligned with the X-axis is the face area times the sin of  $\theta_1$ , as seen in Figure 16.



**Figure 16: Projected Area of a Surface Parallel to X-Axis**

## 2. Atmospheric Attenuation

The atmospheric attenuation, or extinction as it is referred to, is defined by the equation [Ref. 3:p. 4]

$$T1 = \frac{1}{1 + SOG * (e^{\alpha * R} - 1)} \quad (9)$$

where

$T1$  = atmospheric transmission

$R$  = range from sensor to target

$SOG$  = sky-to-ground brightness ratio

$\alpha$  = the extinction coefficient of the atmosphere

This term may be obtained from the base Janus program using the new 1-meter database range. This term is not currently incorporated into ONEMETER so that ONEMETER can be tested and evaluated without the need to access the Janus program.

## 3. Applying Modification Factors

Once all of the modifying factors are known they can be used with the LOS calculation. Recall again Equation 3 from Chapter I,

$$N = CMR * \frac{TDIM}{Range} \quad (10)$$

where

$TDIM$  = target's minimum presented dimension in meters

$Range$  = sensor to target range in kilometers

$CMR$  = Cycles per milliradian

$N$  = number of cycles resolved.

Here is where the ONEMETER program begins to contribute to the overall Janus model.

The target's minimum presented dimension, TDIM, is a one dimensional variable used to describe an amount of a potential target presenting itself for detection. The target defined for the program is composed of surfaces each one square meter in area. It could just as easily be said that each surface has a TDIM of one meter. Therefore for this target one square meter will be equated with a unit length for computational purposes. That is, if the target presents one square meter of area for detection, then the TDIM will be considered to be one meter.

Now, each surface begins with an area of one square meter. As described in section G.1 above this area is first reduced to its projected area. The area is then multiplied by one minus the attenuation factor due to vegetation for further reduction. At this point the atmospheric attenuation would be applied, but as mentioned in section G.2, it is not.

For a given target all calculated areas are then added to produce the net TDIM for the target. This value of TDIM may now be passed to the calling routine.

#### IV. SIMULATION

##### A. DETECTION SIMULATION IN THE ONE METER DATABASE

As discussed, the use of the new one meter database provides for a more detailed and dynamic assessment of target detectability. In order to demonstrate this a small simulation has been run using a small portion of the database and a target moving through ten different locations. The targets locations are shown in Figure 17, labeled T1 through T10. There are a number of obstacle also present in the figure, which correspond to those presented in Figure 3. The object with a 'S' in the center is the small building, the two cross shaped figures with the '8' in them are eight meter tall trees, and the remaining single blocks represent three meter tall trees. The eight meter trees have an under cover index of one meter for each block except the center.

The sensor is placed and remains at grid coordinates (0,0) in the lower left corner. A loop is placed in ONEMETER which steps through each target location, evaluates the LOS, and prints pertinent data to a separate file. This information is contained in Table III.

The first column of the table shows the number of the target being evaluated and the second column shows the range of the target from the sensor. Column three indicates the

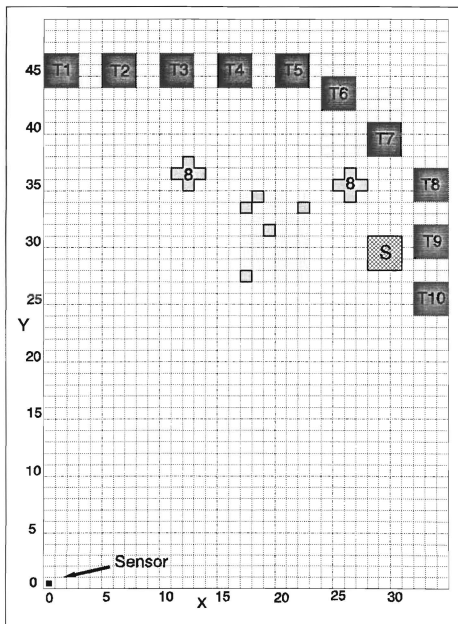


Figure 17: Simulation of Target Detection



Table III SIMULATION DATA

Target Position	Target Range	Number of faces presented	Number of faces seen	Target Area
1	45.01	5	5	3.9984
2	45.40	8	8	4.3377
3	46.32	8	8	4.6816
4	47.76	8	7	4.2380
5	49.66	8	8	5.1713
6	49.74	8	3	2.5761
7	49.41	8	5	3.0118
8	49.58	8	0	0
9	45.97	8	6	4.1171
10	42.64	8	8	5.4574

number of faces of the target that are presented. As discussed in section 3.B, and shown in Figure 6, this value is aspect dependent. The fourth column shows the number of faces that are seen. This is the number of faces to which the sensor actually has some degree of LOS, and is not corrected for the LOS quality. The final column is the total target area visible. This is the sum of each of the faces present in column three modified for area projected and attenuation by foliage. This final value of target area is what will be returned as target minimum presented dimension (TDIM).

## B. EXAMINATION OF A SPECIFIC TARGET

In order to ensure the operation of ONEMETER in this simulation is completely clear the analysis for a single target position will be examined. The target to be studied will be the target located in position T7. As Table III shows, a LOS exists for only five of eight possible faces, and these five LOS's only produce a visible target area of 3.012 square meters. As the analysis of the target is conducted target faces will be referred to as shown in Figure 18.

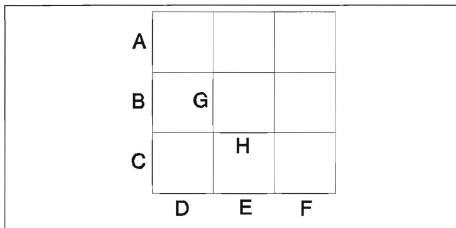


Figure 18 Target Faces

First consider face A, located in the upper left hand corner of the target. Figure 19 shows the pertinent portion of Figure 17 where the target and obstruction come into play. As Figure 19 shows, the LOS passes closely between several trees but reaches the target unobstructed. From trigonometry the LOS impacts face A at an angle of 56.63 degrees from the

normal of the face, and thus only 55 percent of the face area is projected normal to the LOS. The effective area of face A is then

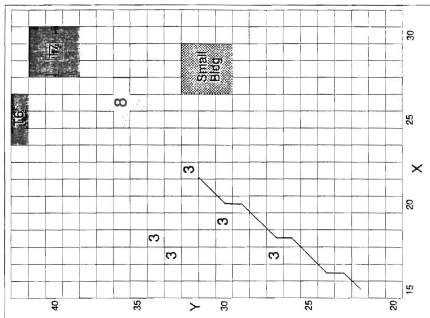
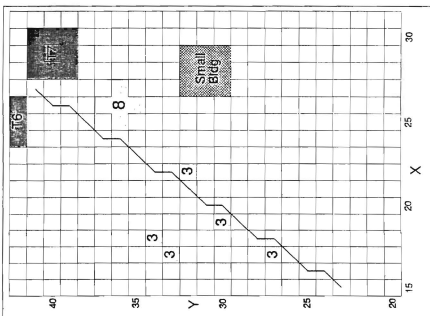
$$1 \text{ meter}^2 * (1 - 0) * 0.55 = 0.55 \text{ meter}^2 \quad (11)$$

where the term (1-0) represents the attenuation of the LOS. A value of one is a perfect LOS, and zero is subtracted because there was no attenuation calculated.

Faces B and C intersect a three meter tree as shown in Figures 20 and 21. The three meter tree is assumed to be solid, and thus no LOS exist to these faces.

The LOS for face D, Figure 22 shows passage through two grids of the foliage of an eight meter tree. However, as Figure 3 in Chapter II shows, the outer grids of the eight meter tree has no branches below one meter in height. Since the sensor is at one meter and face D is assumed to be no more than one meter, the LOS passes below the foliage and is not attenuated. Thus there exists an unobstructed LOS for face D. From trigonometry, and using an equation similar to that above, it can be found that face D has a visible area of 0.805 square meters.

Figure 23 shows the LOS path for face E, which is similar to face F. The LOS passes below the lowest branches of the tree and the LOS is not attenuated. The angle at which the LOS hits face E is slightly different, and as a result 0.795 square meters of face E are visible.



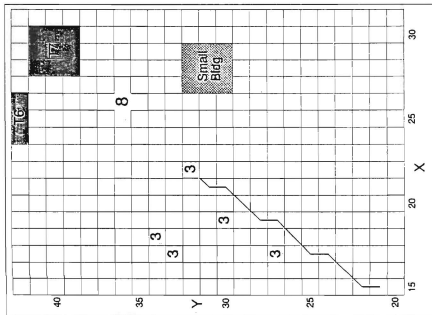


Figure 21: LOS Path to Face C

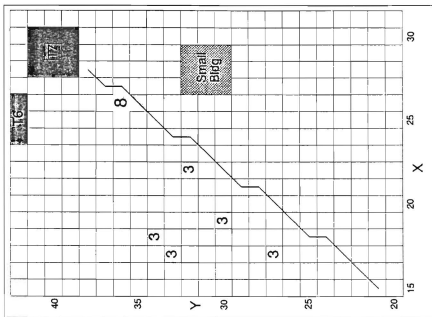


Figure 22: LOS Path to Face D

Figure 24 shows that the LOS for face F avoids all obstacles, and the resulting visible area is 0.785 square meters.

Faces G and H represent the top part of the target shown in Figure 7 of Chapter III. Figure 25 shows that the LOS to face G intersects a three meter tree and is thus blocked entirely.

The LOS to face H is shown in Figure 26 and demonstrates the calculation of attenuation of a LOS. The LOS passes through two grids of and eight meter tree, similar to that shown in Figure 22. However, this time the LOS does not pass below the branches and is attenuated. This is because the upper surface are considered to be at two meters elevation, so the height of the LOS between sensor and target will vary linearly from one to two meters above the ground. Thus the LOS will intersect the foliage in the grid which exists above one meter.

Consider the attenuation resulting from foliage grid intersection. As discussed in section IV.E, this attenuation is a function of distance. Using the assumptions of section E, this function can be written for targets less than 200 meters away as:

$$attenuation = 0.3 * ( 1 + \frac{2.33 * distance}{200} ) \quad (12)$$

The first grid intersected is approximately 42.2 meters away from the sensor, and results in an attenuation of 44.96

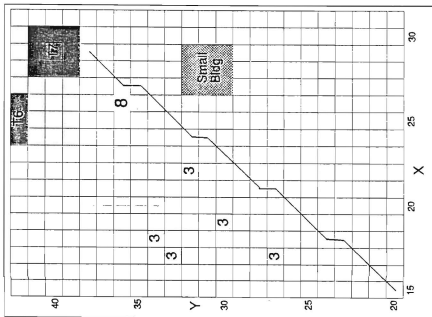


Figure 23: LOS Path to Face 2

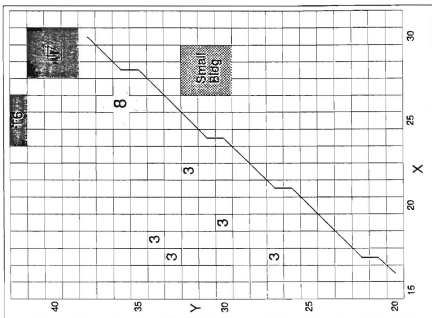


Figure 24: LOS Path to Face F

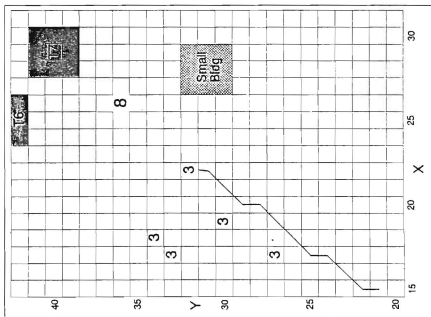


Figure 35: LOS Path CO Face G

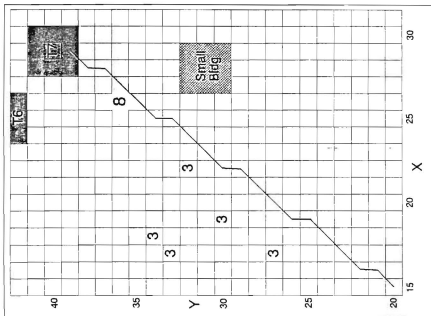


Figure 36: LOS Path CO Face H



percent. The second grid results in an attenuation of 45.45 percent, and when summed a total attenuation of 90.41 percent results.

The area projected normal to the LOS to surface E is 80.3 percent of the full area. Applying the values for face H as was done in Equation 11 yields

$$1 \text{ meter}^2 * ( 1 - .9041 ) * 0.803 = 0.077 \text{ meter}^2 \quad (13)$$

so that only 7.7 percent of face H is visible.

Summing the visible area of all faces for the target results in the total area of 3.012 square meters as indicated in Table III. The target range can be carried out to several decimal places, but this amount of accuracy is probably not necessary for any calculation, a value to the nearest meter should suffice.

### C. DISCUSSION

Note that at several positions some faces of the target are not seen due to obstructing trees. This represents a target in defilade, reducing the visible target area. It is very important to note here that the target's concealment status changes and is recalculated with each different position. This provides more realism than in Janus where the target has only two possible defilade conditions. In Janus a target in defilade is either in partial defilade which reduces the target to one third its size, or full defilade which

reduces the target to one fortieth its size [Ref. 1:p. 135]. Also note that at position 8 the target is behind the building and no LOS exist.

It is the total target area, representing TDIM, which is most significant. In Janus the TDIM is retrieved from the master database and thus remains constant for the target at all times [Ref. 3:p. 6]. However ONEMETER incorporates target aspect and its defilade calculation to create a TDIM which is dynamic as well as precise.

## V. DISCUSSION

### A. EFFECTS OF ONEMETER ON ACQUISITION

#### 1. Replacement of DOLOS

As mentioned in Chapter I, DOLOS is the current subroutine in Janus that determines the existence and quality of the line of sight. The newly developed routine will replace DOLOS in order to use the one meter resolution provided by the Pegasus database.

DOLOS provides the value of PLOS to the subroutines that determine detection and acquisition. Determination of detection at a given time is accomplished by DETECT, found in Appendix H. The subroutine HANDOFF determines if a target already detected can be acquired, and is located in Appendix I. Acquisition is required in order to prosecute a given target.

The value of PLOS from DOLOS is used in DETECT and HANDOFF in a similar manner in each program. In both routines it is necessary to calculate the number of cycles resolved by the sensor on the target. This involves multiplying a target size by the PLOS to get the target's minimum presented dimension, TDIM. Recall Equation 3 from section I.C.3 to see how TDIM is used to find the number of cycles resolved,  $N$ .

$$N = CMR * \frac{TDIM}{Range} \quad (14)$$

DOLOS calculates TDIM for a target as shown below:

$$TDIM = SIZE * PLOS \quad (15)$$

The target size is a fixed value retrieved from the master database, and PLOS is determined as discussed in section D of Chapter I.

It is proposed that ONEMETER will eliminate this calculation providing the total effective area it calculates as the value of TDIM. This will incorporate the new high resolution database and provide more detailed, accurate, and dynamic values of TDIM. This will in turn provide a more realistic value for the number of cycles resolved and the detection and acquisition of targets in Janus.

## B. EFFECT OF ONEMETER ON OTHER SUBROUTINES

### 1. Direct Effects

The program ONEMETER was designed with the intention of improving acquisition in Janus by using the Pegasus database. As discussed above, it is envisioned that ONEMETER will replace the subroutine DOLOS wherever DOLOS is called by DETECT and HANDOFF. However ONEMETER can also be used to benefit any other routines which call DOLOS, thus expanding the usefulness of the one meter database.

The Software Programmer's Manual prepared for Janus contains descriptions of every subroutine that exists in Janus as well as the routines it calls, and those routines which call it. From this it is found that DOLOS is called by 19 different subroutines including DETECT. Table IV lists these routines, gives a brief description of their function, indicates how DOLOS is used in each, and where ONEMETER might be used.

The second column of Table IV indicates subroutines which execute an effective size calculation. In HNRANGE and NRRANGE a component of the calculation involves retrieving the fixed target size from a file and multiplying it by the PLOS and applying a defilade factor. The value of target area provided by ONEMETER replaces this calculation and is a more realistic value as described for DETECT in section A.1.

The routines SFDLOS and SFNRANGE also conduct similar size calculations, but they also consider the size by just using target size and defilade status. ONEMETER output could replace the portions where PLOS is considered, but the portions without PLOS must be addressed. For whatever reason PLOS is not considered, it is not possible to use ONEMETER output because defilade and PLOS are directly related and can not be separated. In order to use the one meter database these subroutines may have to be completely overhauled.

The third column of Table IV indicates the subroutines which use PLOS to calculate another value, such as effective

**Table IV SUBROUTINES THAT CALL DOLOS**

Subroutine	Size	Use PLOS in Calculation	Verify LOS with PLOS	Compare PLOS to 1.0	Task Performed by Subroutine
ADLOS		X			Verifies LOS between a specified radar and a flying target
ADPRODET				X	Determines scan time and detection probability of air defense radar against a flyer
BLUADDET				X	Similar to ADPRODET
DETECT	X	X	X		Determines if target is detected
DFIMPACT				X	Processes direct fire impact event
DISSEV			X		Verifies validity of a guided projectile mission
EVAL			X		Calculates effects of a guided projectile impact
HANDOFF	X	X	X		Determines if acquisition occurs
HORANGE	X	X	X		Construct list of targets within visibility and LOS of a helicopter
KEEPTAR			X		Determines if previously detected target can still be seen after a turn has passed
NORMENR				X	Construct list of enemy flyers in radar tracking range with an LOS
NORANGE	X	X	X		Construct list of targets within visibility and angle limits with LOS
REDADDET				X	Similar to ADPRODET
REDADFIX				X	Set processing delay times for radar based on ability to track target
SFDLOS	X	X	X		Constructs target list for special flyer
SFRANGE	X	X	X		Similar to SFDLOS
SPECRNR				X	Constructs special radar target list
SUSTN			X		Determines status of conventional indirect fire missions
TRAJPRF				X	Verifies no obstacle along a guided projectile flight path

area. All but one of these programs and its use of PLOS have been discussed in the preceding section. The subroutine ADLOS retrieves PLOS for a radar routine, and its exact usage of PLOS is not known.

Column four indicates routines that use PLOS for verification that a LOS exists. In most cases the LOS was previously determined to have existed, and a check is conducted to make sure it still does. PLOS is checked to see that it is greater than a minimum value of 0.1 or 0.0, and as long as it is, then the LOS is assumed to exist for the purpose of the routine. The two exceptions are for SFDLOS and SFRANGE which check that PLOS is greater than 0.9. This would indicate that the special observer examined in the routines requires a high quality LOS.

The fifth column indicates that PLOS is compared to a value of 1.0. If PLOS is not 1.0 then the LOS is not perfectly free of obstructions and is unacceptable for the particular application. The need for a perfect LOS can be understood if the purpose of the subroutine is understood. These subroutines are used either for a flyer or a guided projectile. Because of a flyer's altitude and potentially high speed it would make sense that only a target in the clear could be seen. A guided projectile can not fly through a tree, so if there is any interference it will fail to complete its flight path.

The use of ONEMETER has the potential to benefit all of these subroutines directly by providing more accurate and dynamic values of target size and PLOS.

## **2. Indirect Effects**

Several other subroutines within the Janus structure may also be affected by the implementation of ONEMETER. One program called in DOLOS and discussed previously in section I.D.1, is HITREE. HITREE provided the grid height and density value for DOLOS, and is clearly not needed if DOLOS is done away with. However, HITREE is also called by the routines FASTUP, UNITXYZ, and PEAK. These subroutines should be further examined to ascertain the use of HITREE in each and whether or not they can be modified to use ONEMETER or another new program using the one meter database.

Another routine which should be evaluated for replacement or modification is ASPECT. This is the current routine contained in Janus and should not be confused with the subroutine ASPECT contained within the ONEMETER program. The Janus ASPECT determines the aspect angle for viewing a target. However it only allows three possible aspect angles. Replacement of ASPECT by ONEMETER would probably result in excessive computing time since only an aspect angle is required, but a routine similar to that contained in ONEMETER could be written to provide better evaluation of the aspect angle. Prior to replacement of ASPECT a better understanding



of how aspect is used within the routines that call it is required. The routines that call ASPECT can be found in the description of ASPECT in the Janus Software Programmers Manual.

Determination of other subroutines that may be effected can only be made by gaining a thorough understanding of the operation of Janus. Existing subroutines will require careful examination to determine how best to incorporate values from one meter. Some programs may require no change at all, while others may need significant revision.

### **C. IMPLEMENTATION INTO JANUS**

#### **1. Database Manipulation**

One very important area that has not been addressed is how the Pegasus database will be manipulated and managed for use in Janus. Introduction of Pegasus instantly increases the number of data values by 10000 based solely on the smaller grid size. The increased amount of information available in Pegasus for each grid will also magnify the volume of data. This large amount of information can result in significant slowing of the system if not dealt with efficiently.

For the purposes of ONEMETER, the test battlefield was expanded into its individual components and an array was created to hold each data group. These arrays are structured such that the proper piece of information can be recalled by knowing the X and Y coordinates of the grid. This speeds up

the data recall during the simulation, but would require additional time during the initialization before the scenario is run. The construction of the 32 bit number for the Pegasus database was designed with speed in mind. It is possible to do rapid checks of data values using masks which examine only a single bit of the number and thus speeds up the analytical process [Ref. 5:p. 11].

## **2. Target Representation in Janus**

Another area requiring work in order to use the Pegasus database is target representation. Currently the targets within Janus are represented by several fixed values contained in files of the master database. In order to use ONEMETER and the Pegasus database the targets should be represented in the same style as the Pegasus terrain database. This is necessary because ONEMETER calculates the total visible target area using individual portions of the target. Modifications must also be made, probably in the movement routine, so that the target representation is properly added onto the description of the terrain occupied by the target.

## **3. Three Dimensional Target Surfaces**

One of the key components of the ONEMETER program is the determination of the number of possibly visible faces of the target, and the amount of area projected by each surface normal to the LOS. As currently written, ONEMETER considers only the vertical surfaces of the target. For the target

shown in Figure 7 it there are 13 different vertical one square meter surfaces, but there are also another nine horizontal surfaces. A sensor at an elevation above that of the target has the potential to see the horizontal surfaces on the top of the target. Additionally, if the target happens to be on an inclined surface then all of the surface will have both horizontal and vertical components.

There is clearly a need to analyze all possible surfaces of a target for visibility in calculation of visible area. The solution of this problem is geometrical in nature and could be solved in a number of ways. However, such a complex model is not necessary in order to demonstrate the usefulness of ONEMETER. In addition, since the final method of target representation and implementation onto the terrain is yet unknown, there is no reason to introduce such complexity at this point.

#### **D. VERIFICATION**

##### **1. Program Compatibility with Janus**

ONEMETER is written as a stand-alone program. It does not interact with any other programs and its only interface with anything external is the reading and writing of data from and to files. It has had no interaction with Janus. It is neither called upon as a subroutine nor does it return information to another program.

ONEMETER must be verified to work within the Janus structure. Simply adding it to Janus with a few modifications is not a prudent course of action. The best method of verification would be to construct a program which can call a limited number of Janus routines for use with ONEMETER. This will allow a controlled examination of the response of ONEMETER. The initial host program might simply input a target and sensor location and then call DETECT or HANDOFF to see the response. This clearly points to the need for a thorough understanding of just how the Janus program functions.

## **2. Reality Check**

Once ONEMETER has been properly assimilated into the Janus structure evaluation of its accuracy in approximating actual detection and acquisition can be conducted. This evaluation may even be possible before integration into the full Janus program. During the development of ONEMETER a number of assumptions concerning vegetation density and perception of distant objects were made. These assumptions will remain as only guesses until comparisons to experimentally determined results are made.

Numerous scenarios have already been conducted on the current version of Janus, many of which are supported from data obtained during live field tests. Given the actual data, the proper inputs must be entered into ONEMETER and the

program results must be compared to real results. If a human is unable to detect a specific target obscured by some degree of foliage then the program incorporating ONEMETER should not either. The real data must be used to fine tune ONEMETER.

There are several possible values in ONEMETER which can be adjusted in order to obtain agreement with battlefield experience.

- adjust the value of attenuation for passage through a meter of vegetation, the current attenuation is 30 percent
- change the distance at which an object appears solid, this is currently assumed to be 200 meters
- change the function of attenuation versus distance from linear to some other method such as exponential
- change the manner in which successive attenuation values are combined, currently they are added
- change the value at which the LOS becomes completely lost due to attenuation, current value is 95 percent

Adjustment of any or one of these values will change the visible target area and PLOS for any given sensor-target pair.

## VI. CONCLUSIONS AND RECOMMENDATIONS

### A. CONCLUSIONS

- ONEMETER effectively utilizes the one meter terrain database, Pegasus, to make LOS calculations.
- ONEMETER examines multiple LOS for a single target as opposed to one LOS in the current Janus format.
- Using the detailed information provided by the Pegasus database, ONEMETER can determine whether a potential LOS passes above, below, or through an object.
- ONEMETER through use of the Pegasus database realistically calculates a LOS with the ability to ascertain the effect of small, discrete objects.
- The use of ONEMETER allows for and calculates target properties as they vary with the movement of the sensor unit as well as the target.
- ONEMETER more accurately determines the effective target area through use of multiple LOS, attenuation factors, and projection of target surfaces based on the target aspect angle.
- The improvements in calculating effective target area may be applied to a number of other subroutines within Janus, increasing the overall quality of the program.
- Once its compatibility with Janus is verified, ONEMETER can replace the subroutine DOLOS.

### B. RECOMMENDATIONS

Use of ONEMETER and the Pegasus database have a demonstrated capability to significantly enhance the Janus program and improve realism within evaluation of various scenarios. However further evaluation is required in order to

implement ONEMETER. The areas requiring evaluation and other suggestions are listed below.

- A detailed understanding of how all of the subroutines work and interact with each other must be gained. Then an analysis of how the remainder of the program can benefit from the Pegasus database.
- ONEMETER should be tested to ensure it will operate properly within the Janus programming structure.
- ONEMETER should be evaluated against known detection and acquisition data to verify its accuracy in predicting these events. ONEMETER can then be adjusted to correctly model reality.
- Targets in Janus must be described in the Pegasus database format.
- Subroutines which previously called DOLOS must be modified to call and use ONEMETER.
- A full understanding of exactly how all values in within the Pegasus database structure must be developed and then used to further improve the Janus program through more realistic modeling and speed of calculation.
- Further examine and attempt to incorporate the various factors that affect human perception of objects.

#### LIST OF REFERENCES

1. TITAN Tactical Applications, *Software Programmers Manual Janus(A) 2.1 Model*, Contract No. DABFT 60-90-D-0002, Delivery Order #37.
2. TITAN Tactical Applications, *User Manual Janus 3.0 Model*, Contract No. DABFT 60-90-D-0002, Delivery Order #37.
3. Kellner, A.D., "NVEOD Detection in Janus(A)," memorandum for the record for U.S. Army TRADOC Analysis Command, White Sands Missile Range, New Mexico, 14 July 1992.
4. Celski, R.J., "A Study of the Line of Sight Calculations and Data Base for the Janus(A) Model," paper for TRAC Monterey, Monterey, California, 16 September 1992.
5. Baer, W., and Akin, J.R., "An approach for real-time database creation from aerial imagery," Nascent Systems Development Inc., Carmel Valley, California.
6. Digital Equipment Corporation, Order Number:AA-D034E-TE, *VAX FORTRAN: Language Reference Manual*, Maynard, Massachusetts, June 1988.



14 July, 1992

## MEMORANDUM FOR RECORD

SUBJECT: NVEOD Detection in Janus(A)

## 1.0 PURPOSE

The purpose of this memorandum is to describe portions of the NVEOD detection 'model' which are applicable to the simulation of target acquisition and discrimination in Janus(A). I will attempt to do this in a manner which is understandable by readers who are not necessarily familiar with the details of detection theory, nor with the computer simulation techniques used by Janus(A). However, for the programmer/analyst who might wish to examine (or perhaps modify) the Janus(A) computer code, some relevant Janus(A) subroutines are identified whenever appropriate.

Please note that it is expressly NOT the intent of this memorandum to describe the NVEOD model in its entirety, nor to describe all of the Janus(A) algorithms associated with target acquisition and discrimination in their entirety.

## 2.0 BACKGROUND

The NVEOD detection model is based on a concept involving the computation, for a specific sensor device and target, of the number of 'resolvable cycles' across a target's 'critical' (usually minimum) presented dimension. This concept of resolvable cycles can be visualized as follows:

- a. Imagine a pattern of stripes or bars which are equal in width and alternating in color, positioned at the target's location. Let the contrast between the two colors be the same as the contrast between the actual target's image and its surrounding background. Let the length of the pattern be the same as the target's minimum presented dimension.
- b. Initially, let the width of the stripes be such that an observer with a particular sensor can easily distinguish each individual stripe. Then slowly decrease the width of the stripes until the minimum width at which the observer can still distinguish each pair of stripes is obtained.

- c. Using this minimum width, the number of pairs contained within a distance equal to the target's minimum presented dimension is called the number of "resolvable cycles", or the number of "cycles resolved", for that target and the particular sensor being used.

The concept of resolvable cycles lends itself well to the computation of detection probabilities, as well as to computation of the probability of discriminating a target at a given level. The NVEOD model defines two probabilities associated with detection of a given target by a given sensor:

- PD        - the probability that the target will eventually be detected, given sufficient (infinite) time. This quantity is usually referred to as "P-infinity".
- P(t)      - the probability that the target will be detected during time interval "t", given that the target is within the sensor's field of view, and given that the target can, in fact, eventually be detected.

Both of these quantities, PD and P(t), are functions of the number of cycles "N" which the sensor can resolve across the target's minimum presented dimension. Additionally, the probability that the sensor can discriminate the target at a given level (once the target has been detected) is also a function of N.

Basically, the portion of the NVEOD model relevant to Janus(A) consists of the following three steps:

- a. Calculate attenuation of the target's signature along the line-of-sight between the target and the sensor.
- b. Given the target's signature at the sensor, calculate the number of cycles (N) which the sensor can resolve across the target's critical dimension.
- c. Given N, determine if the target can be detected, and if so, when and at what level of resolution (discrimination).

The following paragraphs describe each of the above three steps in more detail.

### 3.0 ATTENUATION

In Janus(A), we wish to consider the attenuation of a target's signature by the atmosphere, and by an obscurant which we will call 'large-area' smoke. A target's 'signature' is simply some measurable quantity related to that physical attribute of the target which is 'detectable' by a given sensor. (Reflected light and thermal radiation are examples of physical attributes which are detectable by optical and thermal sensors, respectively.)

Let:

St = Signature at target  
Ss = Signature at sensor

Then, in general we may write:

$$S_s = S_t \cdot T_1 \cdot T_2 \quad (1)$$

where

T1 is the 'transmission' of the normal atmosphere.

T2 is the 'transmission' of any large-area smoke along the line-of-sight (LOS) between the target and the sensor.

Note that T1 and T2 are factors which normally take on values between zero and one. Thus, if T1 and T2 are both equal to one, there is no attenuation, and the signature at the sensor is the same as the signature at the target. If either T1 or T2 is equal to zero, the signature at the sensor is equal to zero, and the target cannot be detected by the sensor. When T2 is equal to zero (or less than some minimum threshold value), we say that LOS is 'blocked' by large-area smoke.

For optical sensors, St is the 'optical contrast' of the target. In Janus(A), the optical contrast is part of the 'Weather' data in the master data base, and is therefore the same for all targets during a Janus run.

For thermal sensors, St is defined to be the 'absolute value of the average target-to-background temperature difference', or 'delta-T' of the target. Delta-T is input in the Janus(A) master data base as 'Thermal Contrast Class' for each system (exposed and defilade). Janus internally converts the Thermal Contrast Class to a value of delta-T. (Actually, natural log of delta-T, for reasons which will become clear later.)

For thermal sensors, the atmospheric transmission can be written as:

$$T1 = \text{EXP}(-\text{ALPHA} * R) \quad (2)$$

where

R is the sensor-to-target range.

ALPHA is the "extinction coefficient" of the atmosphere.

For optical sensors, the atmospheric transmission can be written as:

$$T1 = \frac{1}{1 + \text{SOG} * (\text{EXP}(\text{ALPHA} * R) - 1)} \quad (3)$$

where

R is the sensor-to-target range.

ALPHA is the "extinction coefficient" of the atmosphere.

SOG is the "sky-to-ground brightness ratio".

Both ALPHA and SOG are part of the "Weather" data in the Janus(A) master data base. R, of course, is obtained from the target and sensor locations within the simulation.

For computational efficiency, Janus(A) works with the natural log of equation (1) above. Taking the natural log of both sides, equation (1) above becomes:

$$\ln(Ss) = \ln(St) + \ln(T1) + \ln(T2) \quad (4)$$

For the T2 term above, if we define

$$\text{OLEN} = \ln(T2)$$

then equation (4) becomes

$$\ln(Ss) = \ln(St) + \ln(T1) + \text{OLEN} \quad (5)$$

Since the natural log of a transmission is called the "extinction", OLEN is called the extinction due to large-area smoke. Equations for calculation of OLEN will not be presented in this memo.

Note that for thermal sensors, use of equation (2) above allows equation (5) to take on the simple form

$$\ln(S_s) = \ln(S_t) - (\text{ALPHA} * R) + \text{OLEN}$$

which is computed directly whenever needed by subroutine PAIRS in Janus(A). However, the calculation of atmospheric extinction for optical sensors is much more compute-intensive, as can be seen from equation (3) above. For this reason, Janus(A) calculate (during initialization) a table of atmospheric extinction versus range for two optical bands, and stores the results in "slope/intercept" form which allows very fast interpolation of the table values.

Janus(A) routines involved with computation of target signature attenuation are as follows:

```

INITEXT  *Calculates the atmospheric extinction versus
          range tables for optical sensors (two tables,
          one for "Band 1" and one for "Band 2").

PAIRS:   *Directly calculates atmospheric extinction for
          thermal sensors ( - ALPHA * R ).

          *Interpolates the tables build by INITEXT to
          get atmospheric extinction for optical sensors.
    
```

#### 4.0 RESOLVABLE CYCLES (N)

The "performance" of a sensor is represented by a curve of resolvable cycles per milliradian (CMR) as a function of target signature measured AT THE SENSOR. Expressed mathematically:

$$\text{CMR} = f(S_s) \quad (6)$$

where  $S_s$  (as defined earlier) is the target's signature after being attenuated along the LOS ray from the target to the sensor.

No analytical form exists for equation (6), CMR as a function of Ss. Therefore, this curve is input in the Janus(A) master data base (for each sensor) as a table of values. For optical sensors, the quantity Ss above is called 'mean resolvable contrast' (MRC). For thermal sensors, it is called 'mean resolvable temperature' (MRT). Sensor performance is therefore represented in Janus(A) by what is called an MRC or an MRT curve (or table).

As stated earlier, the Janus(A) simulation works with extinction rather than transmissions, for computational efficiency. Using equation (5) above, subroutine PAIRS computes the natural log of the quantity Ss. For this reason, the Janus(A) simulation calculates (once for each sensor) a table of CMR versus the natural log of Ss, and stores the results in a 'slope/intercept' form, which allows very fast interpolation of the table values.

Once CMR is obtained (via table interpolation), the number of cycles resolved by the sensor (N) can be easily calculated from the equation

$$N = CMR \cdot (TDIM/RANGE) \quad (7)$$

where

TDIM is the target's minimum presented dimension (meters),

RANGE is the sensor-to-target range (kilometers).

Actually, the term (TDIM/ RANGE) above is simply a very close approximation for the angle (in milliradians) subtended by the target. TDIM is obtained from the target's 'Minimum Detection Dimension' in the master data base, then modified to take into account the target's defilade status, and partial obscuration of the target by terrain features (if any are present along the sensor-to-target LOS). RANGE is obtained from the sensor and target locations within the simulation.

Major Janus(A) routines involved with computation of resolvable cycles for a sensor-unit/target-unit combination are as follows:

MAKCUR \* Calculates the cycles-per-miliradian versus natural-log-of-target-signature tables (one for each sensor).

GETCPM \* Interpolates MRC/MRT tables from the master data base, for use by MAKCUR.

PAIRS \* Interpolates the tables built by MAKCUR to obtain cycles-per-miliradian (CMR).

\*Calculates the number of miliradians subtended by the target (MST).

\*Calculates the number of resolvable cycles:

$$N = CMR * MST$$

## 5.0 ACQUISITION

Once the number of resolvable cycles has been determined for a particular sensor-unit/target-unit combination, the probability of detection and the distribution of time-to-detect can be easily calculated. Additionally, the level of discrimination (how "well" the observer can see the target) can be determined.

### 5.1 Probability of Detection.

The probability of detection (\*P-infinity\*) is given by NVEOD as

$$PD = (CR ** W) / (1 + CR ** W) \quad (8)$$

where

$$W = 2.7 + 0.7 * CR$$

$$CR = N / N50$$

N is the number of resolvable cycles.

N50 is the median number of resolvable cycles required for eventual detection.

The term \*CR\* above is called the "cycle ratio". N50 is called the "cycle criterion" for detection. Note that when N is equal to N50, CR is equal to one, and equation (8)

evaluates to  $PD = 0.5$ . In other words, for a random sample of observers, half of the observers (on average) will require fewer than N50 resolvable cycles for eventual detection, while half of the observers (on average) will require more than N50 cycles for eventual detection.

In Janus(A), we use the following cycle criteria for detection and subsequent target discrimination:

- |            |  |
|------------|--|
| 1.0 cycles | Detection: Sensing that an object which is foreign to the background is in the sensor's field-of-view.                   |
| 2.0 cycles | Aimpoint: Ability to select an aimpoint on an object which has been determined to be of military interest.               |
| 3.5 cycles | Recognition: Ability to categorize most targets by class, such as tank, APC, truck, etc.                                 |
| 6.4 cycles | Identification: Being able to tell what specific member of a class the target is, for example a T-72 rather than a T-62. |

Note that for a particular sensor-unit/target-unit combination, equation (8) above is ultimately a function of range between the sensor unit and the target unit. (PD is a function of resolvable cycles, which in turn is a function of range.) For a combat simulation such as Janus(A), the range between a sensor unit and a target unit generally changes over time. For example, the initial range between a particular sensor unit and a particular target unit might be very large, with a corresponding very low probability of eventual detection (PD). At some later time, the sensor-to-target range may have decreased to such an extent that the PD is now very large. In order to account for this, the capability of a particular sensor unit to eventually detect a particular target unit must be evaluated periodically. However, we cannot simply recompute PD and compare it to a random draw, each time we wish to re-evaluate the particular sensor/target pair. Doing so is equivalent to considering the particular sensor/target pair to be a different (randomly selected) sensor-unit/target-unit pair. Additionally, the probability that a particular sensor/target pair will pass the PD test becomes a function of how frequently we conduct the test. (In order to see this, ask yourself what would happen if we evaluated each sensor-unit/target-unit pair a million times during each second of simulated game time.)

Janus(A) uses an "initialization" method to account for sensor-to-target range changes in a manner which is not



equivalent to random selection each time an evaluation of a specific sensor-unit/target-unit pair is to be performed. At the beginning of a Janus(A) run, each sensor-unit/target-unit combination receives a random draw from the distribution represented by equation (8) above. These draws represent the number of resolvable cycles required for specific sensor units to be able to (eventually) detect specific target units. These draws (called detection thresholds) are saved, and do not change during a particular scenario run. Janus(A) periodically computes the number of resolvable cycles (N) for a particular sensor-unit/target-unit combination, and compares this number to the previously-stored detection threshold. Whenever N is greater than (or equal to) the detection threshold, we say that the P-infinity test has been passed.

The main Janus(A) routines involved with P-infinity are as follows:

- INITACQ \* Assigns random draws from equation (8) above to every combination of sensor-unit/target-unit contained within the scenario to be executed.
- NRANGE \* Calls subroutine PAIRS to calculate the number of resolvable cycles (N) for a particular sensor/target combination. Compares N to the threshold assigned by INITACQ to determine if the target can eventually be detected.

## 5.2 Time to Detect.

Once the P-infinity test has been passed (the target will eventually be detected), Janus(A) must determine WHEN the target is actually detected.

If we let  $t^*$  be the amount of time that a target has been within a sensor's field-of-view, then the probability that the target will be detected during the time interval  $t^*$  is given by NVEOD as:

$$P(t) = 1 - \exp(-X * T / 3.4) \quad (9)$$

If CR is less than or equal to 2.0, then

$$X = PD$$

else

$$X = CR / 2$$

where CR is the cycle ratio, as defined in paragraph 5.1 above, and PD is the probability of detection (P-infinity) as given by equation (8) in paragraph 5.1 above.

Janus(A) uses equation (9) above to periodically calculate the probability  $P(t)$  that a specific target unit was detected by a specific observer unit during the preceeding time interval  $t$ . A random draw is then compared to  $P(t)$  to determine if the target unit has been detected or not. This process is repeated until the target is detected, or is no longer a candidate for detection by the specific observer unit (i.e. LOS is broken, the target is killed, the observer is killed, etc.)

The main Janus(A) routines involved with  $P(t)$  are as follows:

- PDETEC     \* Calculates  $P(t)$ , given  $t$  and CR. For computational efficiency, PDETEC contains a table of PD vs CR, stored in slope/intercept form for fast interpolation.
- DETECT     \* Calls subroutine PAIRS to calculate the number of resolvable cycles (N) for a particular sensor-unit/target-unit combination. Then calls sub-routine PDETEC to calculate the  $P(t)$  for a specified time interval.

### 5.3 Target Discrimination.

In paragraph 5.1 above we described how Janus(A) uses an 'initialization' method to generate a collection of random draws from the probability distribution represented by equation (8). This distribution is frequently referred to as the P-infinity distribution. Note that the P-infinity distribution has a parameter, N50, whose value is the median value of the distribution.

It is important to realize that we can vary the level of resolution at which a 'detection' takes place, by varying the value of N50 used in generating the P-infinity distribution. For example, if we wish to require that an observer unit be able to categorize a target unit by class (tank, APC, truck, etc.) in order for a 'detection' to occur, then we would use a value of 3.5 cycles for N50 in equation (8). We would then say that we are simulating target acquisition (detection) at the level of 'recognition'. If we use a value of 2.0 cycles for N50 in equation (8), we are simulating target acquisition at the level of 'aimpoint'. It is also important to realize that a collection of random draws, generated by a given value of N50 for the P-infinity distribution, can be 'scaled' to a

collection of random draws generated by the P-infinity distribution for some other value of N50. In other words, if

PD(1.0) represents the P-infinity dist. with N50 = 1.0, and

PD(3.5) represents the P-infinity dist. with N50 = 3.5,

then multiplying each draw from PD(1.0) by the factor 3.5 will give us the equivalent of a collection of draws from PD(3.5).

Janus(A) version 3.0 simulates target acquisition at the level of "aimpoint" (N50 = 2.0 resolvable cycles). During initialization, a random draw (called the "Detection Threshold") from PD(1.0) is stored for each sensor-unit/target-unit combination. This draw is scaled up (by a factor of 2.0) whenever the particular sensor-unit/target-unit combination is being evaluated for (eventual) acquisition. If the scaled-up draw is equal to or greater than 2.0, then the target unit can eventually be acquired by the observer unit.

Once the target has been acquired, the same Detection Threshold draw is (periodically) scaled up by the appropriate factor (3.5 for recognition, 6.4 for identification) to determine if the observer unit can discriminate the target unit at a level higher than aimpoint. The target discrimination level is then used as an input to the direct fire engagement algorithms, and is also used to determine the symbology for graphical display of an enemy unit on a player's workstation screen.

The main Janus(A) routine involved with target discrimination is subroutine DETECT.

*A. D. Kellner*

A.D. Kellner  
Operations Research Analyst  
Janus(A) Development  
Division  
TRAC-WSMR

# APPENDIX B

```

C----- SUBROUTINE--DOLOS ----- A.D.KELLNER, TRAC-WSMR
SUBROUTINE DOLOS ( XS,YS,HS, XT,YT,HT, PLOS )
C-----C
C      PURPOSE:  To determine if line-of-sight (LOS) exists
C      between a "sensor" and a "target", not
C      considering clouds.
C-----C
C      INPUT:
C      XS,YS      - X & Y Map coordinates of "sensor" (Km)
C      HS         - Height of sensor above ground (meters)
C
C      XT,YT      - X & Y Map coordinates of target (Km)
C      HT         - Height of "target" above ground (meters)
C-----C
C      OUTPUT:
C      PLOS       - Probability that LOS exists
C-----C
C      DICTIONARY:
C
C      XSG,YSYG   - X & Y grid-coordinates of sensor
C      XTG,YTGT   - X & Y grid-coordinates of target
C      MAPXY      - array of grid cell data
C      TPLOS      - Scratch used for calc of final PLOS
C-----C
C
C      INCLUDE    'JGLOBE:GLBPARAM.FOR'
C      INCLUDE    'JGLOBE:GLOBTRRN.FOR'
C      INCLUDE    'JGLOBE:GLOBSCR.FOR'
C
C-TCDC  KNT = KNT + 1
C-TCDC  IF( KNT.EQ. 10000 ) THEN
C-TCDC    KOUNT = KOUNT + 10
C-TCDC    PRINT 10010, KOUNT
C-TCDC10010  FORMAT( /, ' --- LOS Count (thousands):', I5 )
C-TCDC    KNT = 0
C-TCDC  ENDIF
C----- Init to no LOS
C
C      PLOS = 0
C----- Compute "Grid-Cell Coordinates" of sensor and target
C
C      XSG = ( XS-XORG ) / GSIZEX
C      YSG = ( YS-YORG ) / GSIZEY
C
C      XTG = ( XT-XORG ) / GSIZEX
C      YTG = ( YT-YORG ) / GSIZEY

```

```

D      IPRNT = 99
D      IF( HS .LT. 0.0 ) THEN
D      HS = ABS( HS )
D      ENDIF

```

C----- GET ELEVATION AT SENSOR LOCATION

```

      IXS = XSG
      IYS = YSG

```

```

      ICELL = IXS + (IYS-1) * IDIMX
      ZS = ( MASKELEV .AND. MAPXY(ICELL) ) + HS

```

C----- Get Elevation & Concealment at TARGET location

```

      IXT = XTG
      IYT = YTG
      ICELL = IXT + (IYT-1) * IDIMX

      ZT = HITREE( MAPXY(ICELL), PL )

```

```

      TPLOS = 1.0
      IF( HT .LT. ZT ) TPLOS = PL * PL

```

```

D      ZZ = ZT
      ZT = ( MASKELEV .AND. MAPXY(ICELL) ) + HT

```

C----- Get Delta-X, Delta-Y in Grid Cells

```

      IDX = ABS( IXT-IXS )
      IDY = ABS( IYT-IYS )

```

```

D      IF( IPRNT .GT. 0 ) THEN
D      TYPE *
D      TYPE *, '----- DOLOS called:'
D      TYPE *
D      TYPE *
D      TYPE *, ' SENSOR Map X,Y,Z =', XS,YS,HS
D      TYPE *, ' TARGET Map X,Y,Z =', XT,YT,HT
D      TYPE *
D      TYPE *, ' SENSOR Grid X,Y,Z =', XSG,YSG,ZS
D      TYPE *, ' TARGET Grid X,Y,Z =', XTG,YTG,ZT
D      TYPE *
D      TYPE *, ' SENSOR Indx IX,IY =', IXS,IYS
D      TYPE *, ' TARGET Indx IX,IY =', IXT,IYT
D      TYPE *
D      TYPE *, '          IDX,IDY =', IDX,IDY
D      TYPE *, '..... T/C hei, PLOS at targt loc =', ZZ,PL
D      TYPE *, '          Init TPLOS =', TPLOS
D      TYPE *
D      ENDIF

```

C----- We will either step in X & compute Y, or  
C step in Y & compute X:

```

      IF( IDY .GT. IDX ) GOTO 500

```

C----- STEP IN X

```

IF( IDX .LT. 2 ) GOTO 800                                ! LOS exists

D  IF( IPRNT .GT. 0 ) THEN
D  TYPE *, ' ----- STEP IN X -----'
D  ENDIF

REALIDX = FLOAT(IDX)

IF( XTG .GT. XSG ) THEN
    ISTART = IXS + 1
    ISTOP  = IXT - 1
    Y      = YSG - 1.0
    DY     = (YTG-YSG) / REALIDX
    Z      = ZS
    DZ     = (ZT-ZS) / REALIDX
ELSE
    ISTART = IXT + 1
    ISTOP  = IXS - 1
    Y      = YTG - 1.0
    DY     = (YSG-YTG) / REALIDX
    Z      = ZT
    DZ     = (ZS-ZT) / REALIDX
ENDIF

CD  IF( IPRNT .GT. 0 ) THEN
CD  TYPE *, ' DY, DZ =',DY,DZ
CD  ENDIF

DO 100 IX = ISTART, ISTOP
    Y = Y + DY
    Z = Z + DZ
    IY = Y

    ICELL = IX + ( IY * IDIMX )
    ZZ = ( MASKELEV .AND. MAPXY(ICELL) )

CD  IF( IPRNT .GT. 0 ) THEN
CD  TYPE *
CD  TYPE *, ' Y, IX,IY =', Y,IX,IY+1
CD  TYPE *, ' Z, ZZ =', Z,ZZ
CD  ENDIF

    IF( Z .LT. ZZ ) GOTO 999                                ! No LOS

    ZZ = ZZ + HITREE( MAPXY(ICELL), PL )

    IF( Z .LT. ZZ ) THEN
        TPLOS = TPLOS * PL
        IF( IPRNT .GT. 0 ) THEN
            TYPE *, '----- ZZ (TREES/CITY) =',ZZ
            TYPE *, '----- PL, TPLOS =', PL,TPLOS
        D  ENDIF
        IF( TPLOS .LT. 0.01 ) GOTO 999                    ! No LOS
        END IF
    END IF

100 CONTINUE

GOTO 800                                                    ! LOS EXISTS

```

[illegible]

```

      PLOS = TPLOS
C----- RETURN to calling routine...

999  CONTINUE

D      IF( IPRNT .GT. 0 ) THEN
D      TYPE *
D      TYPE *, '----- Exit LOS.'
D      TYPE *, '--- PLOS =', PLOS
D      TYPE *
D      TYPE *
D      ENDIF

      RETURN
      END

```



# APPENDIX C

```

C----- FUNCTION--HITREE ----- A.D.KELLNER, TRAC-WSMR
      FUNCTION HITREE ( MAPDAT, PLOS)
C-----
C      PURPOSE:  To return the height of trees or cities
C                from grid data word MAPDAT.
C
C      INPUT:
C
C      MAPDAT    - Terrain grid cell data word
C
C      OUTPUT:
C
C      HITREE    - Local height of trees or buildings (meters)
C
C      PLOS      - Prob of LOS   (obtained from Density code
C                of trees/buildings)
C-----
      INCLUDE      'JGLOBE:GLBPARAM.FOR'
      INCLUDE      'JGLOBE:GLOBTRRN.FOR'

C----- Pull out the density bits

      IDENS = MAPDAT .AND. MASKTREE

      IF( IDENS .EQ. 0 ) THEN
        HITREE = 0.0
        PLOS   = 1.0
        GOTO 999
      ENDIF

C----- Shift IDENS to get correct numerical value

      IDENS = JISHFT( IDENS, -12 )

C----- Pull out the City/Tree type bit

      IF( (MAPDAT.AND.MASKCITY) .EQ. 0 ) THEN
        ITYPE = 1
        ELSE
        ITYPE = 2
      ENDIF
      ! TREES
      ! CITY

C----- Return appropriate value

      INGT = KHGTS(IDENS,ITYPE)
      HITREE = FLOAT( INGT )

C----- If the area is blown down, reduce height
C

```

```

C      IBLOW = MAPDAT .AND. MASKBLOWDOWN
C      IF ( IBLOW .NE. 0 ) THEN
C          HITREE = HITREE / 3
C      ENDIF

C----- Now set density factor

      PLOS = KPLOS(IDENS, ITYPE)
      PLOS = PLOS * 0.01

C----- RETURN to calling routine

999  CONTINUE

D      IF( PLOS .NE. 1.0 ) THEN
D      TYPE *
D      TYPE *, '----- In HITREE:'
D      TYPE *, '--- IDENS, ITYPE =', IDENS, ITYPE
D      TYPE *, '--- HITREE, PLOS =', HITREE, PLOS
D      TYPE *
D      ENDIF

      RETURN
      END

```

**A STUDY OF THE  
LINE OF SIGHT CALCULATIONS  
AND DATA BASE  
FOR THE  
JANUS (A) MODEL**

**MAJOR Robert J. Celski  
TRAC - Monterey  
16 Sep 92**

## I. BACKGROUND

Important in the execution of the Janus (A) model are the line of sight (LOS) calculations. These calculations determine whether there exists LOS between opposing forces; without LOS, direct fire weapons do not engage a target.

Two different factors determine whether LOS exists on the ground:

- Elevation: the elevation level between forces must allow for LOS (i.e., there must not be any masking terrain features between a "sensor" and a "target"). If the intervening terrain between the sensor and the target masks LOS, no direct fire occurs.

- Vegetation and urban features: trees or other types of vegetation and man-made structures naturally interfere with LOS. The Janus (A) model uses both factors to determine LOS for opposing forces. If vegetation and urban terrain features exist on the ground, both must also be accurately represented in the model.

LOS on the ground can be considered as a set of continuous events. When observing over or into a set of trees in the distance, for example, a target may be observed if it is not completely masked by trees or other vegetation. However, other factors such as time available for detection, smoke, weather or good camouflage that may hinder observation. Considering only vegetation and urban factors, modelling LOS to a target is both deterministic and probabilistic, depending on several factors.

The Janus (A) model addresses the *deterministic* nature of observing a target over vegetation and urban terrain by simply calculating whether intermediate terrain masks LOS. The model also addresses the *probabilistic* nature of target detection through vegetation and

urban terrain by considering the density (number and type of trees or buildings per area) of such objects. This is accomplished by assigning discrete density values associated with the probability of "seeing through" each of the different types of vegetation and urban terrain. These density values are further addressed in Section IV.

In order for Janus (A) to accurately model LOS between opposing forces, the vegetation and urban terrain features must be properly represented in terms of heights and densities. *This includes proper representation of the height and density of trees, shrubs, buildings and other man-made structures.* Furthermore, modelling LOS from a sensor to a target lying in a set of trees or buildings can not be continuous (as it actually occurs on the ground); rather, it can be modelled as a set of discrete probabilities. *These probabilities must provide a realistic representation of the LOS into and through the vegetation and urban terrain.*

## II. PROBLEM STATEMENT

The vegetation and urban terrain heights and/or densities may not be properly represented in the Janus (A) model or data base for key areas of interest to TRAC-Monterey analysts. Moreover, the probabilities associated with seeing through the vegetation and urban terrain may be inaccurate, causing misconceptions about what is actually occurring in the model.

## III. OBJECTIVE

The objective of this study is to determine if probabilities of LOS for each of the different vegetation density levels are meaningful. The study only considers vegetation

density levels, since density levels for urban terrain use the same data arrays and calculations as the vegetation densities. Finally, the study attempts to determine if the tree heights corresponding to each density level are adequate in describing the vegetation.

#### IV. DATA

The data used to perform the study is extracted from the Janus (A) terrain data base for a portion of the TEXCOM Experimental Station (TEC), Fort Hunter Liggett, California. Each data point represents 50 square meters of terrain at TEC.

The Janus (A) terrain data base currently uses eight different levels - or *densities* - in determining whether LOS exists. The densities for each 50 square meters of terrain (terrain square) range from level 0, where there is no vegetation - thus no LOS hinderance - to level 7, corresponding to dense vegetation and high tree heights. These vegetation densities for each terrain square are stored in the data base as arrays. Once the density for the terrain square is determined, the tree height and a value for the probability of line of sight (PLOS) into and/or through the vegetation for the entire terrain square is assigned from another array.

A sample array showing terrain squares with the assigned density for each square is shown in Appendix A. This data is stored in a Janus (A) file called terrain.data. The values for tree heights and PLOS corresponding to the eight densities are shown in Table 1. The data used for this table is stored in a data file called terrain810.dat for the vegetation at TEC. A copy of this file is shown in Appendix B.

**TABLE 1 TERRAIN DATA BASE DENSITY LEVELS**

DENSITY LEVEL	TREE HEIGHT (meters)	PLOS <sup>1</sup>
0	0	1
1	3	2
2	7	4
3	9	6
4	10	7
5	11	8
6	13	9
7	14	10

NOTE 1: Except for density level 0, the PLOS in the table represents a relative value, and is converted into an actual probability of line of sight in the program. For density level 0, the PLOS is actually 1 (e.g., there will be LOS when no vegetation exists between sensor and target).

## V. DISCUSSION AND ANALYSIS

The basis for the discussion in this section is the sensor Janus (A) code, written in the FORTRAN language, which is provided in Appendix C. The discussion for tree heights and for the probability of line of sight (PLOS) are each done separately.

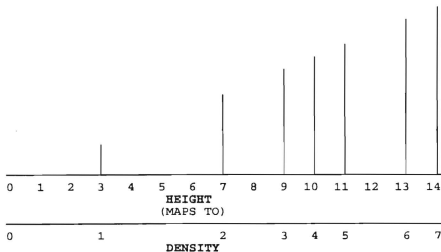
### A. TREE HEIGHTS

Tree heights in Table 1 range from zero meters (no vegetation) in density level zero, to 14 meters in density level seven. However, of the eight density levels, six densities fall at tree levels of seven meters high or above (7, 9, 10, 11, 13, & 14 meters), and only 2 levels are reserved for tree heights below 7 meters (0 & 3 meters). Thus, only 2/7 of all the possible tree heights below seven meters for which a terrain square could be comprised are represented, while 3/4 of those seven meters high and above are represented. For

example, terrain squares having only shrubs of one to two meters in height are misrepresented as having either density level zero or three. Even worse, terrain squares having trees that are five meters in height are misrepresented as having either density three or seven.

Using nine meters as the cutoff, five of the eight densities fall at nine meters or above (9, 10, 11, 13, & 14 meters) - a ratio of 5/6, while only three fall below 9 meters (0, 3, & 7 meters) - a ratio of 1/3. The window of tree heights given the smallest representation lie from one to six meters, where only one of six tree heights are represented.

The tree heights are shown in their actual proportion in Figure 1. The values directly below the x-axis represent tree heights. If there is a line above the x-axis (height) value, the corresponding tree height for the density is represented in the terrain data base in Table 1.



**FIGURE 1** Tree Heights Stored in the Data Base

Note: Not shown is tree height zero; a value of zero is assigned in the data base.



If no line exists (and therefore no density value is assigned to trees of that height) trees of that height are not represented in the data base. The length of the line above the number represents the height of the tree relative to the others for display purposes. The mapping between densities and tree heights is also shown.

The interpretation of this result is that the only tree heights represented in the data base below seven meters are trees that measure three meters in height. The likelihood of this situation is low. There must certainly be terrain squares containing trees whose maximum height is between three and seven meters, yet they are not represented in the data base. Clearly this has an effect on LOS in the model.

## **B. PROBABILITIES OF LINE OF SIGHT (PLOS)**

This sub-section examines two different aspects of PLOS: the data base values assigned for the eight different density levels, and some of the code for PLOS calculations.

### **1. PLOS Data Base**

The third column of Table 1 shows the PLOS assigned to each of the eight different densities. For density level zero, PLOS is always 1.0, and no further calculations are done in the code. The discussion in this paragraph will only cover density levels one through seven.

Table 1 shows that the relative value for PLOS (before further manipulation to transform it into an actual probability) for density level one is two, while the value for density level seven is 10. This is counter-intuitive. If density level zero has the highest PLOS, then the relative PLOS values for increased density levels should be decreasing in order from density level one to seven. This is not what Table 1 shows. Density level seven

represents terrain squares containing much denser terrain, yet the relative PLOS assigned to it is higher than terrain squares assigned a density level of one. Thus, the values for density levels one through seven in the third column of Table 1 appear to be inverted.

## 2. PLOS Calculations

Refer to FUNCTION HITREE (Mapdat, PLOS) in Appendix C for initial code discussions in this paragraph. Below the comment line "C---Now set density factor", PLOS is assigned a value from the KPLOS array (one of the values in column C, Table 1). The next line transforms this value by multiplying the value by 0.01, apparently transforming it into a probability. Thus, neglecting density level zero, the values assigned to the seven remaining densities after this calculation are shown in Table 2.

TABLE 2 PLOS VALUES IN FUNCTION HITREE ( )

DENSITY LEVEL	PLOS
1	0.02
2	0.04
3	0.06
4	0.07
5	0.08
6	0.09
7	0.10

Refer to SUBROUTINE DOLOS ( ) in Appendix D. Below the comment line "C---Get Elevation & Concealment at TARGET location" is the code line invoking FUNCTION

HITREE ( ). The values shown in Table 2 are passed from FUNCTION HITREE ( ) to SUBROUTINE DOLOS ( ). Once passed to SUBROUTINE DOLOS, the PLOS values in Table 2 are temporarily renamed as PL. Two lines below, the value for PL is squared, and renamed as TPLOS (for temporary PLOS), as it will be further used in later calculations. Once the values in Table 2 are squared, the new values for each density (except density zero) are shown in Table 3.

**TABLE 3 INITIAL TPLOS VALUES IN SUBROUTINE DOLOS**

DENSITY LEVEL	TPLOS
1	0.0004
2	0.0016
3	0.0036
4	0.0049
5	0.0064
6	0.0081
7	0.010

The final LOS calculation, based on the densities and discussion in this paragraph, is performed in DO LOOP 100 or 600 in SUBROUTINE DOLOS ( ). Since both loops perform the same calculations on similar data, only DO LOOP 100 is discussed.

There are several calculations performed in DO LOOP 100 which determine if LOS exists between a sensor and a target, first using simple terrain elevation. If there is no masking terrain between sensor and target, the next procedure determines if LOS is blocked by vegetation or urban terrain. Considering only vegetation, another adjustment to TPLOS

is made in the code line written as:

$$TPLOS = TPLOS * PL \quad (1)$$

Recall that PL in SUBROUTINE DOLOS ( ) is the temporary name of the PLOS value returned from FUNCTION HITREE ( ). As a result of this calculation, the values from Table 2 are multiplied by the values from Table 3, yielding new TPLOS values as shown in Table 4.

**TABLE 4** FINAL TPLOS VALUES IN SUBROUTINE DOLOS

DENSITY LEVEL	TPLOS
1	0.000008
2	0.000064
3	0.000216
4	0.000343
5	0.000512
6	0.000729
7	0.001

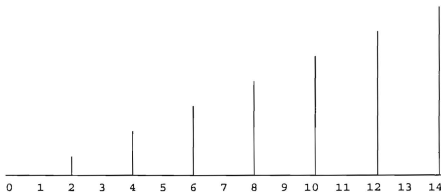
Five lines of code below the calculation in Equation 1, the values in Table 4 are compared to a critical value, to determine if LOS exists. The problem here is that all values in Table 4 lie below the critical value, so the critical value can not possibly be exceeded!! Thus LOS can not possibly exist in this model between a sensor and a target **IF** a target lies in vegetation of any density level other than zero. Clearly, for some non-zero density levels and tree heights, there are cases when LOS exists between a sensor and a target. The code and data contained in the model eliminate this possibility.

## VI. RECOMMENDATIONS AND CONCLUSIONS

The section recommends improvements to the shortfalls addressed above, and concludes with areas for suggested for further study.

### A. RECOMMENDATIONS

The first recommendation addresses tree heights discussed in sub-section V.A. above. Instead of using the distribution of tree heights shown in Figure 1, the tree heights should be represented uniformly from zero to 14. Since there are only eight density levels, a uniform distribution would assign values to every other height, i.e., to zero, two, four,..., 14. Figure 2 represents such a distribution.



**FIGURE 2** Recommended Tree Heights For The Data Base

Note: Not shown is tree height zero; a value of zero is assigned in the data base.

This representation satisfies the need for densities, and is far more accurate in representing the true range of tree heights.

The second recommendation addresses relative values stored in the PLOS data base. For PLOS values entered for density levels one through seven, the entries should be

inverted, where the low density levels have the higher probabilities of LOS assigned to them. Thus lower, less dense vegetation is given a higher value for PLOS than more dense vegetation, and so on.

The third and final recommendation deals with the PLOS calculations. As previously discussed, there is no possible way for LOS to be established if the density level value is other than zero. To more accurately represent reality, areas with shorter, less dense vegetation should allow for LOS. Even dense forests consisting of tall coniferous trees with little underbrush can offer some LOS, and should not be discounted. Thus the data base values in Appendix B listed for each density must be adjusted to allow for some LOS to exist.

## **B. CONCLUSIONS**

To implement the above recommendations admittedly requires much work in the area of data base updating and further testing. However, there are two specific studies that could be performed to correct the problems.

First, a team could be assigned to survey selected areas in the training ranges to estimate tree heights. The number of terrain squares surveyed per square kilometer would naturally depend on the time available for the team. These results could easily be entered in the terrain data base file, and would probably be more accurate than what is available. Furthermore, the recommended tree height distribution discussed above should be used.

Secondly, a more detailed study - probably a thesis - could investigate the true probabilities that should be assigned to density levels one through seven. There are numerous ways to achieve this, such as cluster analysis, and/or determining these

probabilities by simulation. This introduces more reality into the model for line of sight probability of a given vegetation density.

# APPENDIX E

\* Program: CONVERT.F  
 \* this program asks for data used in pegasus database, converts  
 \* that number to binary, builds the total 33 bit binary number  
 \* and converts that number into an integer. To be used for building my  
 \* own battlefield database

```

program transfer
real n
integer m,t(0:32),ele,e12,uci,nor,vgh,vid,nat,ssb,gsv,sum
do 5 i=0,32
  t(i)=0
5 continue
do 20 j=1,9
  if (j.eq.1) then
    m0=0
  elseif (j.eq.2) then
    print*, 'enter gsv (0-64)'
    m0=6
  elseif (j.eq.3) then
    print*, 'enter ssb (0 or 1)'
    m0=7
  elseif (j.eq.4) then
    print*, 'enter nat (0 or 1)'
    m0=8
  elseif (j.eq.5) then
    print*, 'enter vid (0-3)'
    m0=10
    print*, 'enter vgh index (0-15)'
    print*, 'height      index'
    print*, 'water      - 0'
    print*, 'grass       - 1'
    print*, '0-1m        - 2'
    print*, '1-2m        - 3'
    print*, '2-3m        - 4'
    print*, '3-4m        - 5'
    print*, '4-5m        - 6'
    print*, '5-8m        - 7'
    print*, '8-10m       - 8'
    print*, '10-15m      - 9'
    print*, '15-20m     -10'
    print*, '20-25m     -11'
    print*, '25-30m     -12'
    print*, '30-35m     -13'
    print*, '35-40m     -14'
    print*, '40-47m     -15'
  elseif (j.eq.6) then
    m0=14
    print*, 'enter nor (0-15)'
  elseif (j.eq.7) then
    m0=18
    print*, 'enter uci (0-3)'
  elseif (j.eq.8) then
    m0=20
    print*, 'enter hal meter height, 1=.5, 0=0'
  elseif (j.eq.9) then
    m=int(n)
    t(m+m0)=1
    rem=rem-2**m
  goto 10

```



```

endif
20  continue

    sum=0
    do 30 k=0,32
        sum=sum+2**k*t(k)
30  continue

    print*,sum,'=sum'
    print*
    uci=ibits(sum,18,2)
    print*,uci,'=uci'
    print*
    print*,'do you want to try another number? (1=Y, 2=N)'
    read*, a

    if (a.eq.1) then
        go to 1
    endif
end
end

```

# APPENDIX F

```

* Program: BATFIELD.F
  program battle

* This program is designed to place desired obstacles on a playing field
* A variety of objects are offered
  integer db(0:249,0:249)
  open (unit=10,file='field1.dat',status='new')
  ng=1714
  do 5 n=0,249
    do 3 m=0,249
      db(n,m)=ng
    3 continue
  5 continue
  n3=5012

  n7a=8084
*20  n7b=269204

  n10a=9108
  n10b=532372
  n10c=531348

  n15a=10132
  n15b=796564
  n15c=795540
  ns=7208
  n1a=7208
  n1b=8232

  print*,'Welcome to the battlefield builder'
  print*
  print*,'This program will place a variety of objects on a flat,'
  print*,'grass covered, 250x250 meter battlefield. Follow the'
  print*,'instructions of the prompts'
  print*
  print*,'Do you want 3m trees? (1=Y, 2=N)'
  read*,a

*40  if(a.eq.1) then
    print*,'enter the location of tree center (x,y)'
    read*, x,y

    db(y,x)=n3
    print*,'Do you want another 3m tree? (1=Y 2=N)'
    read*,a
    goto 10
  endif
  print*,'Do you want 7m trees? (1=Y 2=N)'
  read*,a

  20  if(a.eq.1) then
    print*,'Enter location of tree center (x,y)'
    read*,x,y
    db(y,x)=n7a
    db(y,x-1)=n7b
    db(y,x+1)=n7b
    db(y-1,x)=n7b
*60

```

```

        db(y+1,x)=n7b
        print*,'Do you want another 7m tree? (1=Y 2=N)'
        read*,a
        goto 20
    endif
    print*,'Do you want 10 meter trees? (1=Y 2=N)'
    read*,a
30  if(a.eq.1) then
        print*,'Enter location of tree center (x,y)'
        read*,x,y
        db(y,x)=n10a
        db(y,x-1)=n10b
        db(y,x+1)=n10b
        db(y-1,x)=n10b
        db(y+1,x)=n10b
        db(y-1,x-1)=n10c
        db(y+1,x+1)=n10c
*80
        print*,'Do you want another 10 m tree? (1=Y 2=N)'
        read*,a
        goto 30
    endif
    print*,'Do you want 15 meter trees? (1=Y 2=N)'
    read*,a
40  if(a.eq.1) then
        print*,'Enter location of tree center (x,y)'
        read*,x,y
        db(y,x)=n15a
        db(y,x-1)=n15b
        db(y,x+1)=n15b
        db(y-1,x)=n15b
        db(y+1,x)=n15b
        db(y-1,x-1)=n15c
        db(y+1,x-1)=n15c
        db(y-1,x+1)=n15c
        db(y+1,x+1)=n15c
*100
        print*,'Do you want another 15m tree? (1=Y 2=N)'
        read*,a
        goto 40
    endif
    print*,'Do you want any small buildings? (1=Y 2=N)'
    read*,a
50  if(a.eq.1) then
        print*,'Enter location of building center (x,y)'
        read*,x,y
        do 70 xx=x-1,x+1
            do 60 yy=y-1,y+1
                db(yy,xx)=ns
60          continue
70        continue
        print*,'Do you want another small building? (1=Y 2=N)'
        read*,a
        goto 50
    endif
*120
    print*,'Do you want any large buildings? (1=Y 2=N)'
    read*,a

```

```

80  if(a.eq.1) then
      print*, 'Enter the location of building center  (x,y) '
      read*, x, y
      do 100 xx=x-2, x+2
          do 90 yy=y-2, y+2
              db(yy, xx) = n1b
90      continue
100     continue
      do 120 xx=x, x+2
          do 110 yy=y, y+2
              db(yy, xx) = n1a
110     continue
120     continue
      print*, 'Do you want another large building?  (1=Y 2=N) '
      read*, a
      goto 80
*140  endif

      do 150 n=0, 249
          do 140 m=249, 0, -1

              write(10, *) db(m, n)

140         continue
150     continue
      end

```

# APPENDIX G

```
* Program: ONEMETER
* this program calculates LOS data on a target moving through the database
* program uses a 250x250 grid
* designed to incorporate attenuation by vegetation
* subroutine aspect, filed as aspect1.f
* is used and determines how many faces of the target are visible, and
* assigns each face a normal direction to be used in the calculation of
* the angle between LOS and face. The main program uses this value
* and determines the total area of the target presented for view, and
* then the total area viewable modified for attenuation and loss of LOS
*****
```

```
program find
```

```
* initialize arrays to hold converted database information
```

```
integer i,xs,ys,xt,yt,tile(0:249999),elev(0:249999)
integer uci(0:249999),vgh(0:249999),vghindex(0:249999),realvgh(0:15)
integer vid(0:249999),nat(0:249999)
integer vistgt(16,4),r,locate(20,2)
```

```
* read database information into array
```

```
open (unit=10,file='field1.dat',status='old')
read (10,*) (tile(i),i=0,62498)
```

```
* open file to output moving target information
```

```
open (unit=11,file='tgtmove1.dat',status='new')
```

```
* assign 1 meter of foliage an attenuation of 30% of the LOS
```

```
denfol = 0.3
```

```
* The vegetation height from pegasus has values of 0-15, each of these
* values represents a particular height or range of heights. The following
* is used to correlate the vgh value to a meaningful height
```

```
realvgh(0)=0
realvgh(1)=0
realvgh(2)=1
realvgh(3)=2
realvgh(4)=3
realvgh(5)=4
realvgh(6)=5
realvgh(7)=8
realvgh(8)=10
realvgh(9)=15
realvgh(10)=20
realvgh(11)=25
realvgh(12)=30
realvgh(13)=35
realvgh(14)=40
realvgh(15)=47
```

```

* Once the database for the grid desired has been read into an array the
* components of information are extracted from the 32 bit number using the
* ibits command. The groups of information of use in the program are placed
* in their own arrays for rapid recall during calculations as follows:

```

```

*
* elevation of highest point in grid - elev
* under cover index - uci
  do 50 i=0,62499

      elev(i)=ibits(tile(i),21,11)
      uci(i)=ibits(tile(i),18,2)
      vghindex(i)=ibits(tile(i),10,4)
      vgh(i)=realvgh(vghindex(i))
      vid(i)=ibits(tile(i),8,2)
      nat(i)=ibits(tile(i),7,1)

```

```

50 continue

```

```

* This program is currently written to run in a stand alone mode.
* Sensor information is entered from the keyboard, and target data
* may be entered from the keyboard or from a data file if several successive
* target locations are desired in order to simulate a moving target.
* The current set up is for a moving target, and the prompts for manual
* target input have been commented out
*
* input grid location and observer height

```

```

      write(11,*),'          xt          yt          target          # faces
      write(11,*)','          range          presented
      write(11,*)'

```

```

* array containing target location, this is used to simulate a moving target

```

```

locate(1,1)=6
locate(1,2)=195
locate(2,1)=14
locate(2,2)=195
locate(3,1)=22
locate(3,2)=195
locate(4,1)=30
locate(4,2)=195
locate(5,1)=37
locate(5,2)=192
locate(6,1)=44
locate(6,2)=186
locate(7,1)=50
locate(7,2)=184
locate(8,1)=56
locate(8,2)=180
locate(9,1)=62
locate(9,2)=176
locate(10,1)=70
locate(10,2)=176
locate(11,1)=78
locate(11,2)=176
locate(12,1)=86
locate(12,2)=176
locate(13,1)=94
locate(13,2)=174
locate(14,1)=99

```

```

locate(14,2)=172
locate(15,1)=101
locate(15,2)=165
locate(16,1)=101
locate(16,2)=157
locate(17,1)=101
locate(17,2)=149
locate(18,1)=101
locate(18,2)=141
locate(19,1)=101
locate(19,2)=133
locate(20,1)=101
locate(20,2)=125

* begin loop inputting targets moving locations
do 900 v=1,20

    xt=locate(v,1)
    yt=locate(v,2)

    range=sqrt(float(xt-xs)**2+float(yt-ys)**2)

* statements to allow manual inputting of target data
*   print*, 'Enter target coordinates (X,Y)'
*   read*, xt,yt
*   print*

* call the aspect subroutine to determine how much of the target
* is presented for possible LOS
* vistgt() is the array returned holding the grid location and information
* on the faces of the target which may be seen.
* n = the number of possible detections and is used for loping the algortihm

    call aspect(xt,yt,xs,ys,vistgt,n)

*loop to check possible LOS for all surfaces presented by target

    r=0

    do 800 q=1,n

* zero atten for each run

        atten=0
        attenf=0

* get the target grid and height data for stepping the LOS

        xt=vistgt(q,1)
        yt=vistgt(q,2)
        ht=vistgt(q,3)

*   print*,xt,' = xt',yt,' = yt'

```

- \* calculate target and observer grid heights, first the height of the ground
- \* must be found by subtracting the vegetation height from the absolute
- \* height. Then the sensor and target heights above ground are added to
- \* obtain absolute elevations of sensor and target

```

zs=elev(xs*250+ys)-vgh(xs*250+ys)
zt=elev(xt*250+yt)-vgh(xt*250+yt)
zs=zs+ho
zt=zt+ht

```

- \* determine the difference between X and Y coordinates, and convert from an
- \* integer to a real number

```

idx=abs(xt-xs)
idy=abs(yt-ys)

rdx= float(idx)
rdy= float(idy)

```

- \* select step in x or y to ensure best grid/slope determination.
- \* it is desired to step towards the target in 1 meter grids along the
- \* longer of the X or Y differences
- \* if idy>idx we skip to stepping in y , otherwise we proceed here

```

if (idy.ge.idx) go to 200

```

- \* determine start and stop grids, slopes for movement direction and elev
- \* the LOS begins in the grid next to the sensor and ends in the LOS
- \* next to the target

```

y=ys
dy=(yt-ys)/rdx
z=zs
dz=(zt-zs)/rdx

```

- \* this if-else statement ensures that we move from the sensor to the target
- \* we only move from the sensor because the distance from the sensor will
- \* affect the level of attenuation of any obstructions

```

if (xt.gt.xs) then
    istart=xs+1
    istop=xt-1
    nn=1
else
    istart=xs-1
    istop=xt+1
    nn=-1
endif

```

- \* apply slopes to each step to determine grid we are passing thru and the
- \* height of the LOS in that grid, compare this height to the height of the
- \* ground, no LOS will exist if ground height > LOS height



```

*
* z = height of LOS
* ztree = height of vegetation
* zdir = height of the ground

ystart=y

do 100 ix=istart,istop,nn

    y=y+dy
    z=z+dz
    iy=nint(y)
    ztree=elev(ix*250+iy)
    zdir=ztree-vgh(ix*250+iy)

* calculate distance from the sensor to the grid in which the LOS is
* currently in as it heads toward target.

    dist=sqrt(float(istart-ix)**2+float(ystart-iy)**2)

* diagnostic statement
* next print statemnet shows blocks thru which los passes
*   print*,ix,iy,vid(ix*250+iy),uci(ix*250+iy),nat(ix*250+iy),z

* compare LOS height to ground height

    if(z.lt.zdir) then
        print*, 'No LOS due to ground intersection'
        go to 800
    else if(z.lt.ztree) then

* the following statements determine attenuation due to vegetation

* if the feature is 200 m away then it appears as a solid object
* this distance has been arbitrarily selected so far

* check the under cover index, if the object has a height, but no* check nature
bit, structures block LOS
* manmade objects have a nature bit of 0

        if(nat(ix*250+iy).eq.0) then
            print*, 'No LOS due to structure'
            go to 800
        endif

* if the program passes the test above then the obstruction is vegetation &
* any other parts of the tree will be assumed as foliage. Current assumption
* is foliage of 1 meter thickness has an attenuation of 30%. This value is
* modified as a function of distance until the modified value reaches an
* attenuation of 100% at the terminal distance, 200meter
* At 200 meters all objects appear solid
*
* it is assumed the modification factor is linear

        if (dist.gt.200) then
            print*, 'No LOS due to distant obstruction'

```

```

        go to 800
        else
            attenf=denfol*(1+2.33*dist/200)
        endif
    endif

* allow a sensor hiding right behind or in the foliage to see thru w/o attenuat'
    if(dist.le.1) then
        attenf=0
    endif

* sum attenuations

    atten = atten + attenf
    print*,atten,'=attenuation'

* if the total attenuation exceeds 95% the LOS is blocked
    if(atten.gt.0.95) then

        print*, 'Loss of LOS due to foliage attenuation'
        go to 800
    endif

100    endif
    continue

* This section is used if we are stepping in the Y direction, and is similar
* to stepping in the X direction

200    print*
        x=xs
        dx=(xt-xs)/rdy
        z=zs
        dz=(zt-zs)/rdy

        if(yt.gt.ys) then
            istart=ys+1
            istop=yt-1
            nn=1
        else
            istart=ys-1
            istop=yt+1
            nn=-1
        endif

        xstart=x

        do 300 iy=istart,istop,nn
            x=x+dx
            z=z+dz
            ix=nint(x)
            ztree=elev(ix*250+iy)
            zdir=ztrees-vgh(ix*250+iy)

* calculate distance

            dist=sqrt(float(xstart-ix)**2+float(istart-iy)**2)

```

```

* diagnostic
*   print*,ix,iy,vid(ix*250+iy), uci(ix*250+iy), nat(ix*250+iy),z
      if(z.lt.zdirt) then
        print*, 'No LOS due to ground intersection'
        go to 800
      else if (z.lt.ztree) then
        if(uci(ix*250+iy).eq.0) then
          print*, 'No LOS due to tree trunk/structure'
          go to 800
        elseif(z.gt.uci(ix*250+iy)) then
          if(nat(ix*250+iy).eq.0) then
            print*, 'No LOS due to structure'
            go to 800
          endif
          if(dist.gt.200) then
            print*, 'No LOS due to distant obstruction'
            go to 800
          else
            attenf=denfol*(1+2.33*dist/200)
            endif
          if (dist.le.1) then
            attenf=0
          endif
          atten = atten + attenf
          print*,atten,'= attenuation'
          if (atten.gt.0.95) then
            print*, 'Loss of LOS due to foliage attenuation'
            go to 800
          endif
        endif
      300 continue
* at this point we have a LOS with atten= 0 or a #
* apply this attenuation factor to the projected area of the target
* to reduce the area of visibility
* projected area is determined based on the surface direction of
* the face in question. This value is 1 for a vertical surface and 2 for
* a horizontal surface. This value is stored in the vistgt array which
* describes the target
      if(vistgt(q,4).eq.0) then
        aproj=rdy/sqrt(rdy**2+rdx**2)

```

```

        else
            aproj=rdx/sqrt(rdy**2+rdx**2)
        endif

* see if there is any attenuation to be applied, if so apply it
        if (atten.eq.0) then

            visaproj=aproj
        else
            visaproj=(1-atten)*aproj
        endif

* sum total visible area presented by the target
        totarea=totarea+visaproj

* zero aproj and visaproj for next LOS
        aproj=0
        visaproj=0

        r=r+1
800    continue

        print*
        print*
        print*,n,' faces of the target present themselves for detection'
        print*,r,' faces of the target are visible to some degree'
        print*,totarea,' square meters of the target are visible'
        print*

* write information to file for moving target
* zero totarea for next run

        write(11,*),xt,yt,range,n,r,totarea

        totarea=0

900    continue

* prompts for single target location entry
* 900    print*,'Do you want to try another set of coordinates?'
*    print*,'Enter 1 for YES, 0 for NO'
*    Read*,a
*    if (a.eq.1) then
*        go to 10
*    endif
*    endif

        end

*****

        subroutine aspect(xt,yt,xs,ys,vistgt,n)

* this subroutine assigns the target grid locations based on the central input
* location. It then uses the sensor and target locations to determine which
* faces of the target are ideally visible to the sensor. Faces which are

```

- \* visible have there information stroed ion the array vistgt() for return to
- \* the main program. The number of faces ideally visible are also returned.

```
real a
integer tgt(16,4),vistgt(16,4),xt,yt,xs,ys
```

- \* assign target data to array tgt() based on center grid of target 3x3
- \* each exterior vertical face of the target is represented

```
tgt(1,1)=xt-1
tgt(1,2)=yt-1
tgt(1,4)=0
tgt(2,1)=xt-1
tgt(2,2)=yt-1
tgt(2,4)=1
tgt(3,1)=xt-1
tgt(3,2)=yt
tgt(3,4)=1
tgt(4,1)=xt-1
tgt(4,2)=yt+1
tgt(4,4)=1
tgt(5,1)=xt-1
tgt(5,2)=yt+1
tgt(5,4)=0
tgt(6,1)=xt
tgt(6,2)=yt+1
tgt(6,4)=0
tgt(7,4)=0
tgt(8,1)=xt+1
tgt(8,2)=yt+1
tgt(8,4)=1
tgt(9,1)=xt+1
tgt(9,2)=yt
tgt(9,4)=1
tgt(10,1)=xt+1
tgt(10,2)=yt-1
tgt(10,4)=1
tgt(11,1)=xt+1
tgt(11,2)=yt-1
tgt(11,4)=0
tgt(12,1)=xt
tgt(12,2)=yt-1
tgt(12,4)=0
tgt(13,1)=xt
tgt(13,2)=yt
tgt(13,4)=0
tgt(14,1)=xt
tgt(14,2)=yt
tgt(14,4)=1
tgt(15,1)=xt
tgt(15,2)=yt
tgt(15,4)=0
tgt(16,1)=xt
tgt(16,2)=yt
tgt(16,4)=1
```

- \* assign target heights, in tgt()

```
do 5 p=1,12
    tgt(p,3)=1
5 continue
do 6 p=13,16
```

```

                                tgt(p,3)=2
6   continue

* establish bounds of the target

    xmax=xt+1
    xmin=xt-1
    ymax=yt+1
    ymin=yt-1

* determine visible sectors of target

    if (xs.le.xmax.and.xs.ge.xmin) then
        if (ys.gt.ymax) then

*           sensor is directly above tgt grids, upper faces

                do 20 j=1,4
                    vistgt(1,j)=tgt(5,j)
                    vistgt(2,j)=tgt(6,j)
                    vistgt(3,j)=tgt(7,j)
                    vistgt(4,j)=tgt(15,j)
                    n=4

*           next lines allowed for skewed view of a top block side

                    if (xs.lt.xt) then
                        vistgt(5,j)=tgt(14,j)
                        n=5

                        elseif (xs.gt.xt) then
                            vistgt(5,j)=tgt(16,j)
20          continue
        else

*           if the sensor is vertically in line with tgt grids and it
*           is not above the tgt it must be below it, sees lower faces

                do 40 j=1,4
                    vistgt(1,j)=tgt(1,j)
                    vistgt(2,j)=tgt(12,j)
                    vistgt(3,j)=tgt(11,j)
                    vistgt(4,j)=tgt(13,j)
                    n=4

*           next lines allow for skewed view of a top block side

                    if (xs.lt.xt) then
                        vistgt(5,j)=tgt(14,j)
                        n=5

                        elseif (xs.gt.xt) then
*           the sensor is horizontally aligned with the tgt grids and
*           to the left of the tgt, sees left side faces

                do 60 j=1,4

```

```

        vistgt(1,j)=tgt(2,j)
        vistgt(2,j)=tgt(3,j)
        vistgt(3,j)=tgt(4,j)
        vistgt(4,j)=tgt(14,j)
        n=4

*       next lines allow for skewed view o a top block side
        if(ys.lt.yt) then
            vistgt(5,j)=tgt(13,j)
            n=5
        elseif(ys.gt.yt) then
            vistgt(5,j)=tgt(15,j)
            n=5
        endif

60      continue

        elseif(ys.lt.ymin) then

*       sensor is to the left and below the target, all of
*       the left side and lower faces seen

        do 80 j=1,4
            vistgt(1,j)=tgt(1,j)
            vistgt(2,j)=tgt(2,j)
            vistgt(3,j)=tgt(3,j)
            vistgt(4,j)=tgt(4,j)
            vistgt(5,j)=tgt(12,j)
            vistgt(6,j)=tgt(11,j)
            vistgt(7,j)=tgt(13,j)
            vistgt(8,j)=tgt(14,j)
80      continue
            n=8
        elseif(ys.gt.ymax) then

*       sensor is to the left side and above the target, all of
*       the left side and upper faces seen

        do 100 j=1,4
            vistgt(1,j)=tgt(2,j)
            vistgt(2,j)=tgt(3,j)
            vistgt(3,j)=tgt(4,j)
            vistgt(4,j)=tgt(5,j)
            vistgt(5,j)=tgt(6,j)
            vistgt(6,j)=tgt(7,j)
            vistgt(7,j)=tgt(14,j)
            vistgt(8,j)=tgt(15,j)
100     continue
            n=8
        endif

        else

*       sensor is right of tgt

        if(ys.ge.ymin.and.ys.le.ymax) then

```

```

*      sensor is horizontally aligned with tgt grids and
*      to the right of the tgt, right faces seen
      do 120 j=1,4
          vistgt(1,j)=tgt(8,j)
          vistgt(2,j)=tgt(9,j)
          vistgt(3,j)=tgt(10,j)
          vistgt(4,j)=tgt(16,j)
          n=4
*
*      the next lines allow a skewed view of a top side
      if(ys.lt.yt) then
          vistgt(5,j)=tgt(13,j)
          n=5
      elseif(ys.gt.yt) then
          vistgt(5,j)=tgt(15,j)
120      continue
      elseif(ys.lt.ymin) then
*
*      sensor is right of and below tgt, all right & lower faces
      do 140 j=1,4
          vistgt(1,j)=tgt(8,j)
          vistgt(2,j)=tgt(9,j)
          vistgt(3,j)=tgt(10,j)
          vistgt(4,j)=tgt(1,j)
          vistgt(5,j)=tgt(12,j)
          vistgt(6,j)=tgt(11,j)
          vistgt(7,j)=tgt(13,j)
          vistgt(8,j)=tgt(16,j)
140      continue
          n=8
      elseif(ys.gt.yamx) then
*
*      sensor is right and above tgt, all right & upper faces seen
      do 160 j=1,4
          vistgt(1,j)=tgt(8,j)
          vistgt(2,j)=tgt(9,j)
          vistgt(3,j)=tgt(10,j)
          vistgt(4,j)=tgt(5,j)
          vistgt(5,j)=tgt(6,j)
          vistgt(6,j)=tgt(7,j)
          vistgt(7,j)=tgt(16,j)
          vistgt(8,j)=tgt(15,j)
160      continue
          n=8
      endif
  endif
  print*
  print*,n,' sectors may be visible'
  print*
  print*, '      xloc      yloc      z      direc'
  print*
  do 180 w=1,n

```



```

*          print*,vistgt(w,1),vistgt(w,2),vistgt(w,3),vistgt(w,4)
* 180 continue
      print*,n,' sectors may be visible'
      return
end

```

# APPENDIX H

C----- SUBROUTINE--DETECT ----- A.D.KELLNER, TRAC-WSMR  
SUBROUTINE DETECT ( IUNIT, ISIDE, CTICK)

PURPOSE: To determine if a target in a given unit's target  
list should be detected now. DSTLOS builds the  
target list.

DICTIONARY:

BARS - Resolvable cycles, sensor's Narrow FOV.  
WBARS - Resolvable cycles, sensor's Wide FOV.  
PROBFOV - Probability that a particular target is  
within the sensor's FOV, at any given time.

INCLUDE 'JGLOBE:GLOBAL.FOR' ! NPEOP  
INCLUDE 'JGLOBE:GLBFLYER.FOR' ! HEIMAST  
INCLUDE 'JGLOBE:GLBSENSR.FOR' ! KSENSCLAS  
INCLUDE 'JGLOBE:GLBMOPPS.FOR' ! FOVMOPP ! CHEM  
  
INCLUDE 'JGLOBE:GLOBUNITS.FOR' ! MOUNTED, TMOVED, KDEFL  
INCLUDE 'JGLOBE:GLOBACQ.FOR' ! Target List, IHAVTARS, KUSESENS  
INCLUDE 'JGLOBE:GLOBDIR.FOR' ! DLEFT, DRIGT  
INCLUDE 'JGLOBE:GLOBLOAD.FOR' ! IAMFIRNG  
INCLUDE 'JGLOBE:GLOBCHEM.FOR' ! CHEM  
INCLUDE 'JGLOBE:GLOBMOVE.FOR' ! IFLYMODE  
INCLUDE 'JGLOBE:GLOBFIR.FOR' ! FIRNXT  
INCLUDE 'JGLOBE:GLBWEAPN.FOR' ! TIMELAY

REAL CONVRT2MIN  
PARAMETER (CONVRT2MIN = 1.0 / 60.0)

REAL FCTR180  
PARAMETER (FCTR180 = 1.0 / 3.14159)

INTEGER\*4 NUMNORFLY  
PARAMETER (NUMNORFLY = 26)

DIMENSION OLENMAX(4)  
DATA OLENMAX / 3.0, 3.0, 7.0, 7.0 /

## TEMPORARY DATA for FLYERS

FLYAOI = Half-angle (radians) for each flyer type's AOI when moving

REAL SIXDEG  
PARAMETER (SIXDEG = 6.0 \* 3.14159 / 180.0) ! FLYER TYPE 3  
REAL PIOVER4  
PARAMETER (PIOVER4 = 3.14159 / 4.0)  
DIMENSION FLYAOI (NUMFLYRS)  
DATA FLYAOI / 2\*PIOVER4,  
SIXDEG, ! Flyer type 3  
29\*PIOVER4 /

```

C
C-----

D      IF( IFIRST.NE.99 ) THEN
D      TYPE *
D      TYPE *, 'Enter DETECT observer unit number for debug print:'
D      ACCEPT *, NUNIT
D      TYPE *, 'Enter DETECT observer SIDE for debug print:'
D      ACCEPT *, NSIDE
D      TYPE *
D      IFIRST = 99
D      END IF

D      IPRINT = 0
D      IF( (IUNIT.EQ.NUNIT) .AND. (ISIDE.EQ.NSIDE) ) IPRINT = 99
D      IF( (NUNIT.EQ.0) .AND. (ISIDE.EQ.NSIDE) ) IPRINT = 99

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *
D      TYPE *, '-----'
D      TYPE *, '***** DETECT *****'
D      TYPE *
D      TYPE *, '--- IUNIT, ISIDE =', IUNIT, ISIDE
CD      ICLOCK = CLOCK / 60.0
CD      ISECS = NINT( CLOCK*60. - ICLOCK*60.0 )
CD      TYPE *, '--- CLOCK (Min/Sec) =', ICLOCK, ISECS
D      TYPE *, '--- CLOCK (Sec) =', CLOCK*60.
CD      TYPE *, '--- IHAVTARS =', IHAVTARS(IUNIT, ISIDE)
D      TYPE *, '--- IAMFIRNG =', IAMFIRNG(IUNIT, ISIDE)
D      TYPE *, '--- FIRNXT =', FIRNXT(IUNIT, ISIDE)*60.
D      ITYPE = KSYSTYP(IUNIT, ISIDE)
D      TYPE *, '--- NPEOP =', NPEOP(ITYPE, ISIDE)
D      END IF

C----- CAN THIS OBSERVER (IUNIT) SEE? -----

      IF( NSCORE(IUNIT, ISIDE) .LT. 1 ) GOTO 999
      IF( KINOPSTAT(IUNIT, ISIDE) .NE. 0 ) GOTO 999 ! CHEM

      IF( MOUNTED(IUNIT, ISIDE) .GT. 0 ) THEN

C----- Unit is a passenger, can he detect targets?

      IHOST = MOUNTED(IUNIT, ISIDE)
      IHTYP = KSYSTYP(IHOST, ISIDE)
      IF( KANHOST(IHTYP, ISIDE) .LT. 2 ) GOTO 999

C----- Yes, update unit's current status

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *, '----- Observer mounted, can search/fire!'
D      ENDIF

      XUNIT(IUNIT, ISIDE) = XUNIT(IHOST, ISIDE)
      YUNIT(IUNIT, ISIDE) = YUNIT(IHOST, ISIDE)
      SPDU(IUNIT, ISIDE) = SPDU(IHOST, ISIDE)
      DVIEW(IUNIT, ISIDE) = DVIEW(IHOST, ISIDE)
      DLEFT(IUNIT, ISIDE) = DLEFT(IHOST, ISIDE)

```

```

        TMOVED(IUNIT,ISIDE) = TMOVED(IHOST,ISIDE)

ENDIF

        ITYPE = KSYSTYP(IUNIT,ISIDE)
        JSIDE = 3 - ISIDE

C----- Do not process special-special flyers if they are popped up.
        IF( ISIDE .EQ. 1
          *   FLYERS(ITYPE,ISIDE) .EQ. NUMFLYRS .AND.
          *   IFLYMODE(IUNIT,ISIDE) .LT. 0 )      .AND.
          *                                           GOTO 999

C----- If one-man system, don't detect new targets while firing/reloading
        IF( NPEOP(ITYPE,ISIDE) .EQ. 1 ) THEN
          IF( IAMFIRNG(IUNIT,ISIDE) .NE. 0 )      GOTO 999
        ENDIF

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *, '--- ITYPE      =', ITYPE
D      TYPE *, '--- IAMFIRNG    =', IAMFIRNG(IUNIT,ISIDE)
D      TYPE *
D      TYPE *
D      TYPE *, '----- OLD TARGET LIST:'
D      TYPE *, '--- Unit Number:', (NMUNIT(J,IUNIT,ISIDE),J=1,NMVISB)
D      TYPE *, '--- Detc Status: ', (KDETECTD(J,IUNIT,ISIDE),J=1,NMVISB)
D      TYPE *
D      TYPE *
D      END IF

C----- Get position of observer ( XO, YO, "first" ZO ) -----
        CALL UNITXYZ ( IUNIT,ISIDE,ITYPE, XO,YO,ZO )

C----- Check if observer is ADA radar -----
        IF( RADARS(ITYPE,ISIDE) .GT. 0 ) THEN

C----- Skip if NORMAL radar
        IF( RADARS(ITYPE,ISIDE) .LE. NUMNORRDR ) GOTO 999

C----- Check for a SPECIAL BLUE or RED Radar handoff
        CALL HANDOFF ( IUNIT, ISIDE, IDIDIT )

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *, '----- NEW TARGET LIST after HANDOFF:'

```

```

D      TYPE *, '--- Unit Number:', (NMUNIT(J,IUNIT,ISIDE),J=1,NMVISB)
D      TYPE *, '--- Detc Status:', (KDETECTD(J,IUNIT,ISIDE),J=1,NMVISB)
D      TYPE *
D      END IF

```

C----- If we have a handoff, point sensor at target

```

      IF( IDIDIT .GT. 0 ) THEN
        PROBOV = 1.0
        GOTO 100
      ENDIF

```

ENDIF

C----- Clear the "handoff accomplished" flag

```

      IDIDIT = 0

```

C----- Get sensor type & class for LOS calculations

```

      ISENS      = KUSESENS(IUNIT,ISIDE)
      ISENS      = KSENSTYP(ISENS,ITYPE,ISIDE)
      ISNSRCLAS  = KSENSCLAS(ISENS)

```

```

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *, '--- ISENS      =', ISENS
D      TYPE *, '--- W-FOV (deg) =', SENSFOV(1,ISENS)*180./3.14159
D      TYPE *, '--- N-FOV (deg) =', SENSFOV(2,ISENS)*180./3.14159
D      TYPE *, '--- ISNSRCLAS  =', ISNSRCLAS
D      ENDIF

```

C----- Modify height of observer ( ZO ) if necessary -----

```

      IFLYTYP = FLYERS(ITYPE,ISIDE)
      IF( IFLYTYP .GT. 0 ) THEN
        ZO = ZO + HEIMAST(IFLYTYP,ISIDE)

```

```

      IF( IFLYTYP .GT. NUMNORFLY ) THEN                                ! SPECIAL flyer

```

```

      IF( IHAVTARS(IUNIT,ISIDE) .EQ. -99 ) THEN                        ! SPECIAL search

```

```

        PROBOV      = 1.0
        ISENS        = KSENSTYP(1,ITYPE,ISIDE)
        ISNSRCLAS    = KSENSCLAS(ISENS)

```

```

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *, '----- SPECIAL flyer search:'
D      TYPE *, '--- XO,YO,ZO      =',XO,YO,ZO
D      TYPE *, '--- ISENS, ISNSRCLAS =', ISENS,ISNSRCLAS
D      TYPE *, '--- PROBOV =', PROBOV
D      END IF

```

GOTO 100

```

      ELSE IF( IHAVTARS(IUNIT,ISIDE) .LT. 0 ) THEN

```

GOTO 999

ENDIF

ENDIF

ENDIF

```
D      IF( IPRINT.NE.0 ) THEN
D      TYPE *, '--- XO,YO,ZO  =',XO,YO,ZO
D      END IF
```

```
C----- CALCULATE PROBFOV -----
C----- PROBFOV is the probability, at any given time, that a particular
C----- target is within the sensor's Field of View (SFOV).
```

```
C----- Fetch current sensor field-of-view (SFOV)
```

```
SFOV = SENSFOV(1,ISENS)                                ! Wide FOV
IF( MOPP(IUNIT,ISIDE) .GT. 0 ) SFOV = SFOV * FOVMOPP      ! CHEM
IF( SFOV .LT. SIXDEG ) THEN                                ! Account for very narrow FOV
  ELEFACTR = SFOV / SIXDEG
  SFOV = SFOV * ELEFACTR
ENDIF
```

```
D      IF( IPRINT.NE.0 ) THEN
D      TYPE *, '--- Actual SFOV (Deg) =', SENSFOV(1,ISENS)*180./3.14159
D      IF( MOPP(IUNIT,ISIDE) .GT. 0 ) TYPE *, '--- FOVMOPP =', FOVMOPP
D      TYPE *, '--- Effctv SFOV (Deg) =', SFOV*180./3.14159
D      END IF
```

```
CC----- Check if Observer is at a PREPO...
```

```
C
C      IF( KDEFL(IUNIT,ISIDE) .LT. 0 ) THEN
C      SEARCH = 2.0 * DLEFT(IUNIT,ISIDE)
C      IF( KDEFL(IUNIT,ISIDE) .EQ. -1 ) THEN                ! Popped-up:
C      PROBFOV = 0.5                                         ! Hueristic to account for Prior Knowledge
C      ELSE
C      ISENS = 2                                             ! Full defilade:
C      ISNSRCLAS = KSENSCLAS(ISENS)                         ! Use sensor 2
C      SFOV = SENSFOV(1,ISENS)                             ! Wide FOV
C      IF( MOPP(IUNIT,ISIDE) .GT. 0 ) SFOV = SFOV * FOVMOPP ! CHEM
C      PROBFOV = SFOV / SEARCH
C      ENDIF
C      GOTO 100
C      ENDIF
```

```
C----- Check if Observer moving or stationary.....
```

```
IF( SPDU(IUNIT,ISIDE) .GT. 0.0 ) THEN
```

```
C----- Observer is moving, do not check Area of Interest,
C----- sensor searches 180 degrees, except flyer searches an AOI assigned
C----- to the particular FLYER type.
```

```

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *, '----- Observer is Moving!'
D      ENDIF

      IF( IFLYTP .GT. 0 ) THEN

        PROBFOV = SFOV / FLYAOI(IFLYTP)

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *, '----- Flyer: Type =', IFLYTP
D      TYPE *, '--- AOI (deg) =', FLYAOI(IFLYTP)*360./3.1416
D      TYPE *, '--- PROBFOV =', PROBFOV
D      END IF

      ELSE

        PROBFOV = SFOV * FCTR180

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *, '----- Non-Flyer:'
D      TYPE *, '--- PROBFOV =', PROBFOV
D      ENDIF

      ENDIF

      ELSE

C----- Observer is not moving,
C      check Area of Interest only if:
C
C      1) observer stationary for more than two minutes, or
C      2) observer is a flyer, or
C      3) observer is in "Pop-up" mode, or
C      4) observer is in full defilade.

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *, '----- Observer is Stationary!'
D      ENDIF

      IF( IFLYTP .GT. 0 ) THEN

        SEARCH = 2.0 * DLEFT(IUNIT,ISIDE)
        PROBFOV = SFOV / SEARCH

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *, '----- Flyer:'
D      TYPE *, '----- Search angle (Deg) =',
D      * SEARCH*180./3.14159
D      TYPE *, '--- PROBFOV =', PROBFOV
D      END IF

      ELSE

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *

```

```

D      TYPE *, '----- Non-Flyer:'
D      ENDIF

      DELT = CLOCK - TMOVED(IUNIT,ISIDE)

      IF( DELT .GE. 2.0 .OR.
*      KDEFL(IUNIT,ISIDE) .LT. 0 .OR.
*      KDEFL(IUNIT,ISIDE) .EQ. 2 ) THEN

C----- Sensor searches AOI only

      SEARCH = 2.0 * DLEFT(IUNIT,ISIDE)
      PROBOFV = SFOV / SEARCH

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *, '--- Stationary more than 2 minutes.'
D      TYPE *, '--- Search angle (Deg) =', SEARCH*180./3.14159
D      TYPE *, '--- PROBOFV =', PROBOFV
D      END IF

      ELSE

C----- Sensor searches 180 degrees

      PROBOFV = SFOV * FCTR180

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *, '--- Stationary less than 2 minutes.'
D      TYPE *, '--- PROBOFV =', PROBOFV
D      ENDIF

      ENDIF

      ENDIF

      ENDIF

100  CONTINUE

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *
D      TYPE *
D      TYPE *,
D      * '----- BEGIN LOOP OVER UNITS IN "NMYUNIT" LIST -----'
D      TYPE *
D      TYPE *, '--- KDEFL (observer) =', KDEFL(IUNIT,ISIDE)
D      TYPE *, '--- ISENS =', ISENS
D      TYPE *, '--- CTICK (Sec) =', CTICK*60.
D      TYPE *, '--- PROBOFV =', PROBOFV
D      TYPE *
D      TYPE *,
D      * '-----'
D      TYPE *
D      END IF

C-----
C----- LOOP OVER TARGET LIST
C----- DO DETECTION CALCULATIONS FOR ALL ENTRIES IN TARGET LIST

```



```

C-----
C----- If we have a radar handoff, ignore other targets

      IF( IDIDIT .GT. 0 ) THEN
            ISTRT = IDIDIT
            IEND = IDIDIT
      ELSE
            ISTRT = 1
            IEND = NMVISB
      ENDIF

      DO 400 NN = ISTRT, IEND

C----- Is slot empty?

            IF( NMYUNIT(NN,IUNIT,ISIDE) .EQ. 0 ) GOTO 400

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *
D      TYPE *, '----- Target unit =', NMYUNIT(NN,IUNIT,ISIDE)
D      END IF

C----- Is the enemy unit still alive? ("JUNIT" is unit number of enemy)

      JUNIT = NMYUNIT(NN,IUNIT,ISIDE)
      IF( NSCORE(JUNIT,JSIDE) .LT. 1 ) THEN
            NMYUNIT(NN,IUNIT,ISIDE) = 0
            KDETECTD(NN,IUNIT,ISIDE) = 0
D      IF( IPRINT.NE.0 ) THEN
D      TYPE *, '--- Target is dead, kick out of list!'
D      END IF
            GOTO 400
      ENDIF

C----- Has the enemy unit mounted?

      IF( MOUNTED(JUNIT,JSIDE) .GT. 0 ) THEN
            NMYUNIT(NN,IUNIT,ISIDE) = 0
            KDETECTD(NN,IUNIT,ISIDE) = 0
            GOTO 400
      ENDIF

C----- Get target unit location

      JTYPE = KSYSTYP(JUNIT,JSIDE)
      CALL UNITXYZ ( JUNIT,JSIDE,JTYPE, XT,YT,ZT )

CD     IF( IPRINT.NE.0 ) THEN
CD     TYPE *, '--- KDETECTD =', KDETECTD(NN,IUNIT,ISIDE)
CCD    TYPE *, '---XO, YO, ZO =',XO,YO,ZO
CCD    TYPE *, '---XT, YT, ZT =',XT,YT,ZT
CD     END IF

C----- Do we still have LOS to the enemy unit?

```

```

CD      IF( IPRINT.NE.0 ) THEN
CD      TYPE *, '----- CHECK FOR LINE-OF-SIGHT : '
CD      END IF

      CALL DOLOS ( XO,YO,ZO, XT,YT,ZT, PLOS )

C----- PLOS IS THE PROBABILITY THAT LINE-OF-SIGHT EXISTS
C----- ( PLOS < 0.0 MEANS SMOKE BLOCKS LINE OF SIGHT )

CD      IF( IPRINT.NE.0 ) THEN
CD      TYPE *, '--- PLOS from "DOLOS" = ', PLOS
CD      END IF

      IF( PLOS .LE. 0.0 ) THEN
          KDETECTD(NN,IUNIT,ISIDE) = 0
          GOTO 400
      ENDIF

C-----
C----- We still have LOS, update detection calculation -----
C-----

      DX = XO - XT
      DY = YO - YT

C-----
C----- COMPUTE TARGET'S EFFECTIVE SIZE -----
C-----

      SIZE = SIZT(JTYPE,JSIDE) * PLOS

CD      IF( IPRINT.NE.0 ) THEN
CD      TYPE *, '--- Actual SIZE = ', SIZT(JTYPE,JSIDE)
CD      TYPE *, '--- PLOS = ', PLOS
CD      TYPE *, '--- SIZE*PLOS = ', SIZE
CD      END IF

C----- Fetch enemy unit's defilade status

      JDEFL = KDEFL(JUNIT,JSIDE)
      JDEFL = ABS( JDEFL )

C----- Check if target in defilade

      IF( JDEFL .GT. 0 ) THEN
          IF( JDEFL .EQ. 1 ) THEN
              SIZE = SIZE * 0.3333333
          ELSE
              SIZE = SIZE * 0.025
          ENDIF
          JCONT = KONCLAS(2,JTYPE,JSIDE)
      ELSE
          JCONT = KONCLAS(1,JTYPE,JSIDE)
      ENDIF

CD      IF( IPRINT.NE.0 ) THEN
CD      TYPE *, '--- After defl, SIZE = ', SIZE
CD      ENDIF

C----- Account for target motion

```

```

      IF( SPDU(JUNIT,JSIDE) .GT. 0.0 ) THEN
        SIZE = SIZE * 1.5
        IF( IPRINT.NE. 0 ) THEN
          TYPE *, '--- Target is moving, SIZE =', SIZE
        ENDIF
      ENDIF

C----- Check if target fired in last 20 Sec
      IF( FLAST(JUNIT,JSIDE) .GE. (CLOCK-0.3333) ) THEN
        PFOV = 1.0

        IF( IPRINT.NE.0 ) THEN
          TYPE *, '----- TARGET FIRED IN LAST 20 Sec!'
          TYPE *, '--- PFOV =', PFOV
        END IF

      ELSE
        PFOV = PROBFOV
      ENDIF

C----- Check if Large Area Smoke (LAS) blocks LOS
      CALL DOLASLOS ( XO,YO,ZO, XT,YT,ZT, ISNSRCLAS, OLEN )

      IF( IPRINT.NE.0 ) THEN
        TYPE *, '----- From DOLASLOS:'
        TYPE *, '--- ISNSRCLAS,OLEN =', ISNSRCLAS,OLEN
      ENDIF

      IF( OLEN .GT. OLENMAX(ISNSRCLAS) ) THEN
        NMYUNIT(NN,IUNIT,ISIDE) = 0
        KDETECTD(NN,IUNIT,ISIDE) = 0
        GOTO 400
      ENDIF

C----- Must consider altitude for flyers as targets or observers
      MODOBS = IFLYMODE(IUNIT,ISIDE) ! Observer

      IF( MODOBS .GT. 0 ) THEN ! Moving/Hover
        IFTYP = FLYERS(ITYPE,ISIDE)
        ZOBS = FLYALTUD(MODOBS,IFTYP,ISIDE)
      ELSE ! Popped Up or on Ground
        ZOBS = - MODOBS
      ENDIF

      MODTGT = IFLYMODE(JUNIT,JSIDE) ! Target

      IF( MODTGT .GT. 0 ) THEN ! Moving/Hover
        JFTYP = FLYERS(JTYPE,JSIDE)
        ZTGT = FLYALTUD(MODTGT,JFTYP,JSIDE)
      ELSE ! Popped Up or Ground
        ZTGT = - MODTGT
      ENDIF

```

```

C----- Convert altitude from meters to KM for range calculation
      ZOBS = ZOBS * 0.001
      ZTGT = ZTGT * 0.001
      DZ = ZOBS - ZTGT

C----- Calculate BARS (resolvable cycles)

      RANGE = SQRT( DX*DX + DY*DY + DZ*DZ )

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *, '-----'
D      TYPE *, '----- CALL PAIRS/PDETEC NEXT, INPUTS:'
D      TYPE *, '--- ISENS, JCONT =', ISENS, JCONT
D      TYPE *, '--- RANGE, SIZE =', RANGE, SIZE
CD     TYPE *, '--- OLEN =', OLEN
CD     TYPE *, '--- CTICK =', CTICK*60.0
D      END IF

      CALL PAIRS ( ISENS, JCONT, RANGE, SIZE, OLEN, BARS )

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *, '--- From PAIRS, BARS =', BARS
D      ENDIF

C----- Check if target already detected

      IF( KDETECTD(NN,IUNIT,ISIDE) .GT. 0 ) THEN

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *, '----- Target previously detected:'
D      TYPE *, '--- CHECK IF CLOUDS BLOCK LOS:'
D      TYPE *
CD     CALL PRESSRET
D      END IF

      IF( RANGE .GT. 0.025 ) THEN

          CALL SMOKELOS ( XO,YO,ZO, XT,YT,ZT, ISNSRCLAS, ISEE )

          IF( ISEE .EQ. 0 ) THEN
              IF( IPRINT.NE.0 ) THEN
                  TYPE *
                  TYPE *, '--- CLOUDS NOW BLOCK LOS!!!!!!!!!!'
                  CALL PRESSRET
                  ENDIF
                  KDETECTD(NN,IUNIT,ISIDE) = 0
                  GOTO 400
              ENDIF
          ENDIF

          GOTO 300

      ENDIF

      NO, See if we can detect this target.....

      WBAR = BARS * CNVRTCPM(ISENS)

```

```

CALL PDETEC ( WBARS, CTICK, PD )

PD = PD * PFOV
CALL UNIRAN ( DRAW )

D      IF( IPRINT.NE.0 ) THEN
D      CALL PDETEC ( WBARS, CTICK, PPD )
D      TYPE *, '--- WBARS, PFOV =', WBARS, PFOV
D      TYPE *, '--- Pure PD, Eff-PD =', PPD, PD
D      TYPE *, '--- DRAW =', DRAW
D      ENDIF

      IF( DRAW .GT. PD ) GOTO 400          ! Not detected

D      IF( IPRINT.NE.0 ) THEN
D      TYPE 763, JUNIT
D 763   FORMAT ( //, '!!!!!!!!!!!!!! TARGET',
D      *      I4, ' MAY BE DETECTED !!!!!!!!!!!!!', //)
D      TYPE *, '--- BY UNIT, SIDE =', IUNIT, ISIDE
D      TYPE *
D      TYPE *, '--- CHECK IF CLOUDS BLOCK LOS:'
D      TYPE *
D      ISNSRCLAS = -ISNSRCLAS      ! Only if FORD done for SMOKELOS
D      CALL PRESSRET
D      ENDIF

      IF( RANGE .GT. 0.025 ) THEN

          CALL SMOKELOS ( XO,YO,ZO, XT,YT,ZT,
D      *      ISNSRCLAS, ISEE )

          IF( ISEE .EQ. 0 ) THEN
D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *, '--- CLOUDS BLOCK LOS!!!!!!!!!!'
D      CALL PRESSRET
D      ENDIF
D      GOTO 400
          ENDIF

      ENDIF

      KDETECTD(NN,IUNIT,ISIDE) = 1
D      IF( MOVERS(JTYPE,JSIDE) .EQ. 3 )
D      *      KDETECTD(NN,IUNIT,ISIDE) = 2

D      IF( IPRINT.NE.0 ) THEN
D      TYPE 767, JUNIT
D 767   FORMAT ( //, '!!!!!!!!!!!!!! TARGET',
D      *      I4, ' IS DETECTED !!!!!!!!!!!!!', //)
D      TYPE *, '--- BY UNIT, SIDE =', IUNIT, ISIDE
D      TYPE *, '--- CLOCK =', CLOCK*60.
D      TYPE *, '--- IAMFIRNG, FIRNXT =',
D      *      IAMFIRNG(IUNIT,ISIDE), FIRNXT(IUNIT,ISIDE)*60.
D      CALL PRESSRET
D      ENDIF

D      CALL WDETEC ( IUNIT,ISIDE, JUNIT,JSIDE,
D      *      KUSESENS(IUNIT,ISIDE) )

```

```

C----- If it is not currently firing,
C----- schedule a potential direct fire event for this unit...

      IF( IAMFIRNG(IUNIT,ISIDE) .GT. 0 ) GOTO 300

C----- Fetch weapon to use

      CALL WTCHWPN ( IUNIT,ITYPE,ISIDE, JTYPE, RANGE,
*                               IWSLOT, IWPN )

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *, '--- Weapon to use:   IWPN =', IWPN
D      TYPE *
D      ENDIF

      IF( IWPN .LE. 0 ) GOTO 300

C----- Schedule fire event, based on Aim and Lay Times for the weapon

      TLAY = TIMELAY(IWPN,ISIDE)           ! LAY TIME
      TAIM = TIMEFIR(IWPN,ISIDE)           ! AIM TIME
      TAIM = TAIM * CONVRT2MIN

      IF( TLAY .GT. 0.0 ) THEN
        FIRNXT(IUNIT,ISIDE) = CLOCK
      ELSE
        TLAY = (-TLAY) * CONVRT2MIN
        IF( MOPP(IUNIT,ISIDE) .EQ. 1 )
*          TLAY = TLAY * EFFICMOPP(IWPN,ISIDE)
        FIRNXT(IUNIT,ISIDE) = CLOCK + TLAY
      ENDIF

C----- Add aim time

      FIRNXT(IUNIT,ISIDE) = FIRNXT(IUNIT,ISIDE) + TAIM

C----- Re-schedule RELOAD, if necessary

      IF( FIRNXT(IUNIT,ISIDE) .LT. TNEXT(4) ) THEN
        NXTUNIT = IUNIT
        NXTSIDE = ISIDE
        TNEXT(4) = FIRNXT(IUNIT,ISIDE)
      ENDIF

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *, '----- Set next firing time!'
D      TYPE *, '--- TAIM (Sec) =', TIMEFIR(IWPN,ISIDE)
D      TYPE *, '--- TLAY (Sec) =', TIMELAY(IWPN,ISIDE)
D      TYPE *, '--- CLOCK (Sec) =', CLOCK*60.
D      TYPE *, '--- FIRNXT (Sec) =', FIRNXT(IUNIT,ISIDE)*60.
D      TYPE *
D      CALL PRESSRET
D      ENDIF

C----- Target is detected (either just now, or previously).
C----- See if we can increase its Detection Level.

```

CONTINUE

```

      IF( KDETECTD(NN,IUNIT,ISIDE) .LT. 3 ) THEN
        IVAL = KACQPERF(JUNIT,IUNIT,ISIDE)
        IF( KDETECTD(NN,IUNIT,ISIDE) .EQ. 1 ) THEN
          DRAW = PAIRSVAL(IVAL) * 3.5           ! For Recognition
          IF( BARS .GE. DRAW ) THEN
            KDETECTD(NN,IUNIT,ISIDE) = 2
            DRAW = PAIRSVAL(IVAL) * 6.4         ! For Identification
            IF( BARS .GE. DRAW ) THEN
              KDETECTD(NN,IUNIT,ISIDE) = 3
            ENDIF
          ENDIF
        ELSE
          DRAW = PAIRSVAL(IVAL) * 6.4           ! For Identification
          IF( BARS .GE. DRAW ) THEN
            KDETECTD(NN,IUNIT,ISIDE) = 3
          ENDIF
        ENDIF
      ENDIF

D      IF( IPRINT.NE.0 ) THEN
D      IVAL = KACQPERF(JUNIT,IUNIT,ISIDE)
D      DRAW = PAIRSVAL(IVAL)
D      TYPE *
D      TYPE *, '----- Final:'
D      TYPE *, '--- Acq Perf (draw):'
D      TYPE *, '--- Recognition    =', DRAW*3.5
D      TYPE *, '--- Identification =', DRAW*6.4
D      TYPE *, '--- BARS           =', BARS
D      TYPE *, '--- KDETECTD      =', KDETECTD(NN,IUNIT,ISIDE)
D      TYPE *
D      ENDIF

400  CONTINUE

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *, '----- END LOOP OVER UNITS IN "NMUNIT" LIST -----'
D      TYPE *
D      END IF

999  CONTINUE

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *, '----- End of DETECT.'
D      TYPE *
D      END IF

      RETURN
      END

```

# APPENDIX I

```

C----- SUBROUTINE--HANDOFF ----- A.D.KELLNER, TRAC-WSMR
SUBROUTINE HANDOFF ( IUNIT, ISIDE, IDIDIT )
  INCLUDE 'JGLOBE:GLOBAL.FOR'
  INCLUDE 'JGLOBE:GLOBUNITS.FOR' ! KSYSTYP,RADARS,FLYERS
  INCLUDE 'JGLOBE:GLBSNSR.FOR' ! KSENSCLAS
  INCLUDE 'JGLOBE:GLOBACQ.FOR' ! Target List
  INCLUDE 'JGLOBE:GLBFLYER.FOR' ! FLYALTUD
  INCLUDE 'JGLOBE:GLOBMOVE.FOR' ! IFLYMODE

  DIMENSION OLENMAX(4)
  DATA OLENMAX / 3.0, 3.0, 7.0, 7.0 /

CD IF( IFIRST.NE.99 ) THEN
CD TYPE *
CD TYPE *, 'Enter HANDOFF observer unit number for debug print:'
CD ACCEPT *, NUNIT
CD TYPE *, 'Enter HANDOFF observer SIDE for debug print:'
CD ACCEPT *, NSIDE
CD TYPE *
CD IFIRST = 99
CD END IF

CD IPRINT = 0
CD IF( (IUNIT.EQ.NUNIT) .AND. (ISIDE.EQ.NSIDE) ) IPRINT = 99
CD IF( (NUNIT.EQ.0) .AND. (ISIDE.EQ.NSIDE) ) IPRINT = 99
D IPRINT = 99

D IF( IPRINT.NE.0 ) THEN
D TYPE *
D TYPE *
D TYPE *, '-----'
D TYPE *, '***** HANDOFF *****'
D TYPE *
D TYPE *, '--- IUNIT, ISIDE =', IUNIT,ISIDE
D TYPE *
D END IF

  IDIDIT = 0

  ITYPE = KSYSTYP(IUNIT,ISIDE)
  JSIDE = 3 - ISIDE

C----- Get sensor type & class for LOS calculations

  ISENS = KUSESENS(IUNIT,ISIDE)
  ISNS = KSENS(TYP,ISENS,ISIDE)
  ISNSRCLAS = KSENSCLAS(ISENS)

CD IF( IPRINT.NE.0 ) THEN
CD TYPE *
CD TYPE *, '--- ITYPE, ISENS, ISNSRCLAS =', ITYPE,ISENS,ISNSRCLAS
CD TYPE *, '--- RADARS(ITYPE,ISIDE) =', RADARS(ITYPE,ISIDE)
CD END IF

C----- Get position of observer ( XO, YO, ZO ) -----

```



```

CALL UNITXYZ ( IUNIT,ISIDE,ITYPE, XO,YO,ZO )

CD      IF( IPRINT.NE.0 ) THEN
CD      TYPE *, '--- XO,YO,ZO  =',XO,YO,ZO
CD      END IF

C-----
C                      LOOP OVER LONG TARGET LIST
C-----

CD      IF( IPRINT.NE.0 ) THEN
CD      TYPE *
CD      TYPE *, '----- LOOP OVER UNITS IN "KLONGTLU" LIST -----'
CD      TYPE *
CD      TYPE *, '--- ITARLIST  =', MYTARLIST(IUNIT,ISIDE)
CD      TYPE *, '--- WPNRNG    =', PRIMRNG(ITYPE,ISIDE)
CD      TYPE *
CD      END IF

      WPNRNG      = PRIMRNG(ITYPE,ISIDE)
      WPNRNGSQR   = WPNRNG * WPNRNG

      ITARLIST    = MYTARLIST(IUNIT,ISIDE)

      DO 400 NN = 1, NUMVISBL

C----- Is enemy detected and ready for track (KLONGTLS .EQ. 2)

      IF( KLONGTLS(NN,ITARLIST) .LT. 2 ) GOTO 400

C----- Is slot empty?

      IF( KLONGTLU(NN,ITARLIST) .EQ. 0 ) GOTO 400

C----- Is the enemy unit still alive? ("JUNIT" is unit number of enemy)

      JUNIT = KLONGTLU(NN,ITARLIST)
      IF( NSCORE(JUNIT,JSIDE) .LT. 1 ) THEN
         KLONGTLU(NN,ITARLIST) = 0
         KLONGTLS(NN,ITARLIST) = 0
         GOTO 400
      ENDIF

C----- Get target unit location

      JTYPE = KSYSTYP(JUNIT,JSIDE)
      CALL UNITXYZ ( JUNIT,JSIDE,JTYPE, XT,YT,ZT )

C----- IUNIT is a radar, determine altitude for target only

      MODE = IFLYMODE(JUNIT,JSIDE)

      IF( MODE .GT. 0 ) THEN
         JFTYP = FLYERS(JTYPE,JSIDE)
         DZ     = FLYALTUD(MODE,JFTYP,JSIDE)
         ! Moving/Hover
      ELSE
         ! Popped Up or On Ground

```

```

        DZ = - MODE
    ENDIF

C----- Convert altitude from meters to kilometers

    DZ = DZ * 0.001
    DX = XO - XT
    DY = YO - YT

C----- Is this target within max weapon range?

    RANGESQR = DX*DX + DY*DY + DZ*DZ

CD      IF( IPRINT.NE.0 ) THEN
CD      TYPE *
CD      TYPE *, '-----'
CD      TYPE *, '----- FOR ENEMY TARGET UNIT =', JUNIT
CD      TYPE *, '-----'
CD      TYPE *
CD      TYPE *, '--- KLONGTLS =', KLONGTLS(NN,ITARLIST)
CD      TYPE *
CD      TYPE *, '--- XO, YO, ZO =', XO,YO,ZO
CD      TYPE *, '--- XT, YT, ZT =', XT,YT,ZT
CD      TYPE *, '--- RANGE =', SQRT( RANGESQR )
CD      END IF

    IF( RANGESQR .GE. WPNRNGSQR ) GOTO 400

C----- Do we still have LOS to the enemy unit?

CD      IF( IPRINT.NE.0 ) THEN
CD      TYPE *
CD      TYPE *, '--- CHECK FOR LINE-OF-SIGHT : '
CD      END IF

    CALL DOLOS ( XO,YO,ZO, XT,YT,ZT, PLOS )

C----- PLOS IS THE PROBABILITY THAT LINE-OF-SIGHT EXISTS
C----- ( PLOS < 0.0 MEANS SMOKE BLOCKS LINE OF SIGHT )

CD      IF( IPRINT.NE.0 ) THEN
CD      TYPE *, '----- PLOS FROM "DOLOS" =', PLOS
CD      END IF

    IF( PLOS .LE. 0.0 ) GOTO 400

C-----
C----- We still have LOS, is the target detectable -----
C----- Compute target's effective size

    SIZE = SIZT(JTYPE,JSIDE) * PLOS * 1.75           ! 3.5 / 2.0

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *, '--- Actual SIZE =', SIZT(JTYPE,JSIDE)
D      TYPE *, '--- PLOS =', PLOS
D      TYPE *, '--- Eff SIZE =', SIZE

```

```

D      END IF

C----- Fetch target's thermal contrast

JCONT = KONCLAS(1,JTYPE,JSIDE)
CALL DOLASLOS ( XO,YO,ZO, XT,YT,ZT, ISNSRCLAS, OLEN )

IF( OLEN .GT. OLENMAX(ISNSRCLAS) ) THEN
    KLONGTLU(NN,ITARLIST) = 0
    KLONGTLS(NN,ITARLIST) = 0
    GOTO 400
ENDIF

RANGE = SQRT( RANGESQR )

CD      IF( IPRINT.NE.0 ) THEN
CD      TYPE *
CD      TYPE *, '-----'
CD      TYPE *, '----- CALL PAIRS/PDETEC NEXT, INPUTS:'
CD      TYPE *, '--- ISENS =', ISENS
CD      TYPE *, '--- JCONT =', JCONT
CD      TYPE *, '--- RANGE =', SQRT( RANGESQR )
CD      TYPE *, '--- SIZE =', SIZE
CD      TYPE *, '--- EFFTIM =', TEFFTIM*60.CD      TYPE *, '--- OLEN =',
OLEN
CD      END IF

      CALL PAIRS ( ISENS, JCONT, RANGE, SIZE, OLEN, BARS )

CD      IF( IPRINT.NE.0 ) THEN
CD      TYPE *
CD      TYPE *, '--- OUTPUT BARS =', BARS
CD      IVAL = KACQPERF(JUNIT,IUNIT,ISIDE)
CD      TYPE *, '--- BARS NEEDED =', PAIRSVAL(IVAL)*2.0
CD      END IF

      IVAL = KACQPERF(JUNIT,IUNIT,ISIDE)
      AIMPOINT = 2.0 * PAIRSVAL(IVAL)
      IF( BARS .LT. AIMPOINT ) GOTO 400

      IF( RANGE .GT. 0.025 ) THEN
          CALL SMOKELOS ( XO,YO,ZO,XT,YT,ZT,ISNSRCLAS, ISEE )
          IF( ISEE .EQ. 0 ) THEN

CD              IF( IPRINT.NE.0 ) THEN
CD              TYPE *
CD              TYPE *, '--- CLOUDS BLOCK LOS!!!!!!!!!!'
CD              CALL PRESSRET
CD              END IF

                  KLONGTLU(NN,ITARLIST) = 0
                  KLONGTLS(NN,ITARLIST) = 0
                  GOTO 400

      ENDIF
      ENDIF

```

```

        WPNRNGSQR = RANGESQR
        IDIDIT = NN

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *, '----- SET IDIDIT =', IDIDIT
D      TYPE *
D      END IF

        GOTO 500      ! Handoff first tgt which meets all requirements

400    CONTINUE

CD     IF( IPRINT.NE.0 ) THEN
CD     TYPE *
CD     TYPE *, '----- END LOOP OVER UNITS IN "KLONGTLU" LIST -----'
CD     TYPE *
CD     END IF

C----- Do we have a target to handoff to the (Normal) Target List?

500    CONTINUE

        IF( IDIDIT .GT. 0 ) THEN

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *
D      TYPE *, '----- Old Short Tgt List:'
D      DO 977 I = 1, NMVISB
D      TYPE *, I, NMYUNIT(I, IUNIT, ISIDE), KDETECTD(I, IUNIT, ISIDE)
D 977    CONTINUE
D      TYPE *
D      ENDIF

        JUNIT = KLONGTLU(IDIDIT, ITARLIST)

C----- Clear Target List, but keep old detected status if present

        IAMSEEN = 0
        DO 600 NN = 1, NMVISB
            IF( NMYUNIT(NN, IUNIT, ISIDE) .EQ. JUNIT ) THEN
                IAMSEEN = KDETECTD(NN, IUNIT, ISIDE)
            ENDIF
            NMYUNIT(NN, IUNIT, ISIDE) = 0
            KDETECTD(NN, IUNIT, ISIDE) = 0
600    CONTINUE

C----- Put target in first slot

        IDIDIT = 1
        NMYUNIT(IDIDIT, IUNIT, ISIDE) = JUNIT
        KDETECTD(IDIDIT, IUNIT, ISIDE) = IAMSEEN

D      IF( IPRINT.NE.0 ) THEN
D      TYPE *
D      TYPE *
D      TYPE *, '-----'

```

```

D      TYPE *, '----- HANDOFF to short target list:'
D      TYPE *, '--- SLOT      =', IDIDIT
D      TYPE *, '--- NMYUNIT    =', NMYUNIT(IDIDIT, IUNIT, ISIDE) ^G^G[22H
D      TYPE *, '--- KDETECTD   =', KDETECTD(IDIDIT, IUNIT, ISIDE)
D      TYPE *, '-----'
D      TYPE *
D      END IF

```

```

      ENDIF

```

```

999    CONTINUE
D      TYPE *
D      TYPE *
D      TYPE *, '----- EXIT HANDOFF.'
D      TYPE *
D      TYPE *
D      RETURN
D      END

```

## TARGET ACQUISITION IN JANUS ARMY

## 1.0 INTRODUCTION

1.1 Purpose. This paper describes the target acquisition function in Janus Army. Particular attention is paid to stating the pertinent assumptions and identifying the rationale for the algorithms and methods.

1.2 Scope. The basic detection process of visual and thermal sensors versus ground targets is emphasized at the expense of the special code required for radars, special purpose sensors, and flying targets. These special cases will be described in later work.

## 1.3 Background.

a. Target acquisition in Janus Army is based on heuristics and theory which have been loosely collected and are known as the search model of the Army's Center for Night Vision and Electro-Optic (CNVEO) devices.<sup>1, 2</sup> The model computes the single-glimpse probability by considering the number of resolvable cycles across the target (the Johnson criteria)<sup>3</sup>; the number of resolvable cycles needed for detection, aim-point, recognition, and identification is a function of the clutter level and is empirically determined input to the model.<sup>4</sup> The model has been adapted for use in several combat simulations including Janus Army, and the assumptions and limitations of the basic CNVEO search model must be considered in addition to those of each specific adaptive implementation. A summary of the basic model as used in Janus Army is also provided at enclosure 1 as a complementary description.

b. Janus Army is interactive, and the requirement for rapid response to the gamers' commands strongly influences the implementation of the CNVEO search model. Pre-processing is designed to reduce the computation required for search processing, and the search is partitioned so that not all units are processed for detection every cycle. Filters are included to avoid unnecessary trips through computationally expensive algorithms. These trade-offs in design may have obscured some of the mathematics and rationale of the algorithms from a reader of the Janus Army code. It is one goal of this paper to provide a more accessible explanation.

c. The search function in Janus Army is implemented in three parts. In the first, initialization, the parameters are set up to partition the search process as a function of the numbers of units on each side and two time constraints input by the gamer. Also, the sensor performance level for the population of observers is initialized for every observer-target pair in the game, and the

sensor performance characteristics by sensor type are initialized. In the second part, units which meet the appropriate criteria described in section 3 below are placed on a list as potential targets. In the third part, the probability of detection by specific observers of targets from the potential target list is calculated and random draws are made to determine which targets are actually detected.

## 2.0     INITIALIZATION

### 2.1     Search Process Control

a. The search and line-of-sight (LOS) processes are relatively heavy users of computational resources. Trade-offs must be made between the frequency of search updates and the responsiveness of gamer interactions. The process is designed to allow the gamer to specify a target list update cycle that is longer than the detection cycle so that units may be considered less often for the filtering process and computation may be saved at the expense of a somewhat less accurate list of potentially detectable targets. The user is allowed to specify the cycle times for the two processes during the run initialization, and the process then constructs the search loop parameters from the numbers of blue and red units and the two input time constraints.

b. Janus screen I allows the specification of the target list cycle time and the detection cycle time in seconds. These values are stored by subroutine GETSCRN into the variables DTDSTLOS and DTSEARCH, respectively. The subroutine INITSENS calculates the search loop parameters KSRCHINC and KSRCHLOOP.

c. The calls for the detection cycle are relatively straightforward. Subroutine SEARCH calls subroutine DODETECT which calls subroutine DETECT for every unit on one side having potential targets. Each side is considered on alternating calls to DODETECT, so that every unit on the potential target list is considered for detection on every two calls to subroutine SEARCH. Subroutine INITSENS takes the initial value of DTSEARCH in seconds, converts to minutes, and divides by 2 to allow two calls to SEARCH by subroutine RUNJAN so that the detection cycle occurs in the time specified by the user.

d. The construction of the potential target list is also driven from the calls to SEARCH through calls to subroutine DODSTLOS. DODSTLOS calls subroutine DSTLOS for particular units in sequences of length KSRCHINC, which is set by subroutine INITSENS to be  $3 \leq KSRCHINC \leq 10$ . The units are filtered for inclusion as potential targets in batches of KSRCHINC until one side is completed, then the other side is processed in batches of KSRCHINC. Subroutine INITSENS calculates the number of batches required for all units to be processed in this fashion within the time interval DTDSTLOS consistent with the calls to SEARCH. This number is KSRCHLOOP and is used in SEARCH so that DODSTLOS is

called KSRCHLOOP times for each call to SEARCH. This integral partitioning means that the period of the target list cycle, for a given unit, expressed in the game time interval between calls to such subroutines as DSTLOS or NRANGE, is only approximately equal to the input value specified by the user.

e. Examples of the process for determining the search control parameters are given in this section. Assume the user has specified a detection cycle of 9 seconds and a target list cycle of 20 seconds. Thus, RUNJAN calls SEARCH every 4.5 seconds and a given unit on the potential target list is considered for detection every 9 seconds. If there are 100 blue units and 200 red units in the game, the subroutine INITSENS will initialize KSRCHLOOP to be 12 and KSRCHINC to be 5. These parameters will produce the following pattern in potential target processing.

TABLE 1. FIRST EXAMPLE OF PARTITIONING FOR SEARCH

CALL TO SEARCH	POTENTIAL TARGET PROCESSING FOR
1	60 BLUE UNITS
2	40 BLUE UNITS AND 20 RED UNITS
3	60 RED UNITS
4	60 RED UNITS
5	60 RED UNITS

This means that a given unit will be considered every 5 calls to SEARCH or every 22.5 seconds, a bit longer than the 20 seconds specified. The partition is not always so even, as the next example shows. Assume the target list cycle to be 40 seconds instead of 20. The algorithm gives KSRCHLOOP = 6 and KSRCHINC = 5, producing the following processing partition.

TABLE 2. SECOND EXAMPLE OF PARTITIONING FOR SEARCH

CALL TO SEARCH	POTENTIAL TARGET PROCESSING FOR
1	36 BLUE UNITS
2	36 BLUE UNITS
3	28 BLUE UNITS AND 6 RED UNITS
4	36 RED UNITS
5	36 RED UNITS
6	36 BLUE UNITS
7	36 BLUE UNITS
8	36 RED UNITS
9	14 RED UNITS AND 18 BLUE UNITS

Note that not every search cycle necessarily considers the same number of units. No more calls to DSTLOS are made within a batch after the last unit on one side is processed; the processing for



the next side does not start until the next call to DODSTLOS starts a new batch. The above example shows that all units are processed within 9 calls to SEARCH or within 40.5 seconds, although some have a shorter interval of 36 seconds because the partition does not come out perfectly even.

f. The choice of the detection and target list cycles can certainly influence the sensor performance within a given Janus game. Shorter cycles will generally provide a better representation of the search process at the price of increased run time and slower interactive response for a given scenario.

## 2.2 Sensor Performance Capability

a. For a particular ensemble of observers with unlimited time to search for a target, there will generally be a subset of observers who will never find a target with a given size, range, and contrast.<sup>1</sup> The fraction of observers who can find the target is called p-infinity (PINF). The distribution of the number of resolvable cycles each member of a normal observer ensemble needs for target detection is characterized by the cumulative probability PINF which is assumed to be

$$\text{PINF} = R \cdot E / (1 + R \cdot E) \quad (1)$$

where  $R = N/N50$  is the ratio of the number  $N$  of resolvable cycles across the target to the number  $N50$  of cycles required for PINF to be 0.5, and  $E = 2.7 - 0.7 (N/N50)$ .<sup>2</sup>

b. In Janus-Army, the detection capability of the set of observers is assumed to be characterized by the capability of each observer to detect each target given unlimited time. This capability is initialized in subroutine INITACQ for each observer-target pair and remains constant throughout a game. The criterion  $N50$  for detection has been given by CNVEO<sup>5</sup> to be 0.5 to 1.0, and 1.0 is the value currently used. The current implementation of the cumulative distribution uses solutions to equation (1) with  $N50 = 1.0$  to precompute values of PINF corresponding to 100 intervals of width 0.01. The initialization process stores a randomly generated pointer to the PAIRSVAL table which contains the value of the resolvable cycles required for the appropriate PINF reflecting the detection capability for the given sensor-target pair. This provides a threshold determining whether a given sensor will eventually detect a given target providing a given number of cycles (or greater) are resolvable.

2.3 Detection Process. The probability of detection as a function of time is given<sup>1</sup> by the standard equation

$$P(t) = 1 - \exp(-t/\text{TAU}) \quad (2)$$

where TAU is the mean time to detect. Two expressions have been given<sup>1</sup> as approximations for the time factor  $1/\text{TAU}$ . For  $N/N50 \leq 2$ ,

$$1/\text{TAU} = \text{PINF}/3.4. \quad (3)$$

This approximation gives probabilities which are too small for the most obvious targets, those with  $N/N50$  greater than 2. For these values,

$$1/\text{TAU} = N/(6.8 \cdot N50). \quad (4)$$

The empirical justification for the estimates in equations (3) and (4) is discussed in detail by Lawson, et. al.<sup>1</sup> In summary, the approximation of  $1/\text{TAU}$  by  $\text{PINF}/3.4$  or  $N/(6.8 \cdot N50)$  was based on field test results.

For speed of computation, the time factor  $1/\text{TAU}$  is represented as a function of the resolvable cycles  $N$  by constructing a set of straight lines from point to point as  $N$  varies from 0 to 4.5 in steps of 0.1125. Equations (3) and (4) are used to calculate the points for  $N50 = 2$ . The results are used to generate slope and intercept data for the straight lines between each neighboring pair of points, and the data are set in variables  $\text{SLOPE}$  and  $\text{CEPT}$  in subroutine  $\text{PDSTEC}$ . This routine scales the resolvable cycles to produce results equivalent to equations (3) and (4) for  $N50 = 1$  (detection).

**2.4 Optical Extinction.** Subroutine  $\text{INITEXT}$  initializes the maximum ranges of optical sensors in two frequency bands and initializes the linear interpolation approximation to the extinction curve. The quantity called "extinction" in the comments in the Janus code is called "contrast transmittance" in some other references.

a. The standard equation for contrast is used to find the range at which the optical contrast is reduced to the minimum detectable contrast ( $\text{CONMIN} = 0.02$ ) for the eye. The contrast available at a range  $R$  from a target is

$$C(R) = [L_T(R) - L_B(R)]/L_B(R), \quad (5)$$

where  $L_T(R)$  is the luminance of the target and  $L_B(R)$  is the luminance of the background.<sup>6</sup> The target and background luminances at range  $R$  can be written

$$L_T(R) = L_T(0) T_A(R) + L_P(R) \quad (6)$$

$$\text{and} \quad L_B(R) = L_B(0) T_A(R) + L_P(R) \quad (7)$$

where  $T_A(R) = \exp(-\alpha R)$  is the transmittance at range  $R$  through the atmosphere with extinction coefficient  $\alpha$  according to Beer's law, and  $L_P(R)$  is the extraneous energy scattered or

remitted into the sensor field of view over the distance  $R$ .<sup>7</sup> From equations (5), (6), and (7),

$$C(R) = C(O) / [1 + (L_p(R) / L_b(O)) / T_a(R)]. \quad (8)$$

For optical paths from sensor to target near the horizon,

$$L_{sky}(R) = L_{sky}(O) T_a(R) + L_p(R), \quad (9)$$

but if it is assumed that the atmosphere is a homogeneous natural atmosphere, the sky is just as bright at  $R$  as it is at the target, so that

$$L_p(R) = L_{sky} [1 - T_a(R)]. \quad (10)$$

Let  $L_{sky}/L_b(O)$  be  $S_g$ , the sky-to-ground ratio, then equation (8) becomes

$$C(R) = C(O) / [1 - S_g (1/T_a(R) - 1)]. \quad (11)$$

Representative estimates of  $S_g$  are provided by Hoock<sup>6</sup> and Ruschke<sup>8</sup> relating solar elevation (with clear skies) to visibility for two surface reflectances (0.15 and 0.30). Let  $F = C(R)/C(O)$  be the contrast transmittance (or extinction), then from (11),

$$R = \frac{1}{\alpha} \log \left[ \left( \frac{1}{F} - 1 + S_g \right) / S_g \right] \quad (12)$$

The maximum range  $R_m$  for the given frequency band is taken from equation (12) for  $C(R) = \text{COMIN} = 0.03$  as implemented in subroutine INITEXT. No optical contrast is available beyond  $R_m$  for each of two optical frequency bands.

b. From equation (12), the contrast transmittance is

$$F = 1 / [1 + S_g (\exp(\alpha R) - 1)]. \quad (13)$$

A curve of  $\log F$  versus  $R$  is constructed in slope-intercept form in subroutine INITEXT. The slopes are stored in OEXTCURVE(1, I, IBAND) and the intercepts in OEXTCURVE(2, I, IBAND) with IBAND equal to 1 or 2, and I ranging from 1 to  $N_R + 1$  where  $N_R$  is the number of range points. The log of equation (13) is used to make the linear interpolation a better approximation.

c. It is the contrast transmittance, or extinction, that is implemented in Janus Army, not the contrast itself, but users should be aware of the implications of definitions of contrast. Note that the contrast as defined by equation (5) assumes that the target is brighter than the background or else contrast could be negative. All optical contrasts input to Janus Army are positive. This is not a necessary restriction, and contrast is frequently

defined as the absolute value of the difference in target and background luminance divided by the background luminance,

$$C(R) = |L_T(R) - L_B(R)| / L_B(R). \quad (14)$$

This is considered a more usual definition for our purposes, and the equations (12) and (13) of extinction are still valid.

d. Both equations (5) and (14) are still open to the objections of contrast being undefined for  $L_B(R) = 0$ , such as would be the case for a target against an overcast night sky. Another definition which is more general and would handle such cases has been stated<sup>5</sup> as

$$C(R) = |L_T(R) - L_B(R)| / \text{Max} [L_T(R), L_B(R)] \quad (15)$$

This definition has no problem with a non-luminous background ( $L_B = 0$ ) but requires some modifications to the extinction equations. For  $L_T(R) \geq L_B(R)$ , equations (12) and (13) are still valid, but for  $L_B(R) > L_T(R)$ , equations (12) and (13) need to be replaced by

$$R = (1/\alpha) \log \{ [1/F - 1 - (1 - C(0)) S_g] / [1 - C(0)] S_g \} \quad (16)$$

$$\text{and } F = 1 / \{ 1 + [1 - C(0)] S_g [\exp(\alpha R) - 1] \}. \quad (17)$$

respectively. In effect, the sky-to-ground ratio  $S_g$  must be modified by the factor  $1 - C(0)$  when the background is brighter than the target in order for the extinction to be consistent with the definition of contrast in equation (15). So far, there has not been a need to use contrast as defined in equation (15) in Janus Army.

e. Recent work considers variations in the background. For some definition such as

$$\text{CONTRAST} = \frac{\text{TARGET LUMINANCE} - \text{LOCAL BACKGROUND LUMINANCE}}{\text{AVG BACKGROUND LUMINANCE}}$$

the equations (12) and (13) involving extinction would still apply as implemented in Janus Army. The implementation of contrast in Janus Army is a good approximation for current applications, but users wishing to extend Janus Army into new target environments need to determine the definition of contrast underlying the input data.

2.5 Thermal Contrast. In Janus Army targets are placed into twelve contrast classes according to the temperature difference (degrees Celsius) between the target and the background. The logarithms of the appropriate temperature differences are stored as data in subroutine PAIRS, and the input contrast class is used as an index.

TABLE 3. CONTRAST CLASSES

CONTRAST CLASS	TEMPERATURE DIFFERENCE, $\Delta T$ (Degrees C)	$\log \Delta T$
1	0.5	0.693147
2	1.0	0.0
3	1.5	0.405465
4	2.0	0.693147
5	2.5	0.916291
6	3.0	1.098612
7	3.5	1.252763
8	4.0	1.386294
9	4.5	1.504077
10	5.0	1.609438
11	6.0	1.791759
12	7.0	1.945910

2.6 Sensor Performance. Janus Army expects sensor data in terms of cycles resolved across the target per milliradian of target area subtended, for a given level of contrast. The contrast is just the optical contrast for sensors using the visible frequency bands and is the absolute value of temperature difference for thermal sensors. Subroutine MAKCUR is called from INITSENS to generate an interpolation curve for cycles per milliradian versus the logarithm of the contrast value for each type of sensor.

### 3.0 POTENTIAL TARGET LIST

The Janus Army detection process has a first stage of filtering units to avoid the search and line-of-sight (LOS) calculations for as many units as possible. The units are processed in batches as described in section 2.1.

3.1 Unit Characteristics. Subroutine DSTLOS contains most of the logic and bookkeeping for filtering units before they can be added to the potential target list. The unit is skipped whenever it is destroyed, inoperable due to chemical, or is a passenger and cannot detect targets.

3.2 Unit Data. Subroutine SETOBSRV is called by DSTLOS to set up the unit data needed for the search and detection routines. The location of the observer and the view-fan parameters are found. (The view-fan is constructed by the gamer and is symmetric about the direction of view.) The input view-fan is used unless the unit is a ground vehicle which is moving or has been stationary less than two minutes, in which case the unit is assumed to search 360°. This assumption is designed to reduce the burden of setting and adjusting view-fans for coordinated surveillance by groups of vehicles while moving, and it provides

approximately the same effect for a group of vehicles as would individually set and coordinated view-fans. The type sensor to use for the next search interval is determined. For each observer, the maximum visibility is used to construct a square centered at the observer's location to serve as a filter for possible targets. Each unit's target list is checked for previous detections, and subroutine KEEPTAR is called to check if a previously detected target can still be seen. KEEPTAR checks for destroyed targets, targets mounted inside vehicles, visibility limits, presence in the view-fan, LOS, and whether large area smoke clouds block LOS. Previously detected targets which can still be seen are flagged, and the observer's target list is cleared.

### 3.3 Visibility, LOS, and Detectability Filters

a. Subroutine NRANGE is called by subroutine DSTLOS to construct the potential target list. For a particular observer unit, enemy units which pass the filters in this subroutine are placed on the potential target list. Note that friendly units are not considered as targets. The filters in NRANGE are very similar to those in KEEPTAR; the main difference is that the sensor performance is tested in NRANGE for the capability to eventually recognize this target.

b. The enemy unit is ignored for detection if it is dead or is mounted inside another unit (a vehicle). The location of the enemy unit is compared to the visibility limits and to the view-fan (called area of interest). The targets are examined for how well they can be seen as represented by the number of resolvable cycles the target presents to the sensor. The target size is adjusted to account for the effects of various signatures. A discussion of threshold scaling can be found near the end of the enclosed memorandum. Multiplying a target size by a factor is equivalent to dividing the acquisition threshold N50 by the same factor.

(1) Targets that are in defilade present a smaller target size to the observer. For partial, or hull defilade, the exposed part of the target is assumed to be one third (0.333333) of the full size. For full defilade, the exposed part of the target is assumed to be one fortieth (0.025) of full size. These factors are based on the relative sizes of turrets and sights that might typically be exposed on common vehicles.

(2) The effect of movement on the process of target acquisition has been long debated. In the past, the cycle criterion for a moving vehicle target was assumed to be half of that required for a stationary target.<sup>9</sup> Recent versions of Janus did not have any signature change due to target motion. The latest release of Janus Army, Version 3.0, contains an inference from recent work with moving human targets,<sup>9</sup> that detection of moving vehicles should also be somewhat easier than of stationary

ones. The size of moving targets in Janus Army 3.0 is scaled by a factor of 1.5 as an estimate of this effect.

(3) For detection of walking or standing human targets, an appropriate cycle criterion from recent field tests would be between 0.30 and 0.50.<sup>9</sup> Such effects are approximated in Janus Army by doubling the size of human targets.

c. For sensors with a zoom capability, version 3.0 of Janus Army contains a more explicit representation of sensor field-of-view (FOV) than did previous versions. A conversion factor CNVRTCPM (for each sensor type) has been added to the data base to allow the approximation of wide FOV performance. The sensor performance data in the Janus(A) data base are appropriate for narrow FOV. The sensor performance for wide FOV is obtained by multiplying the cycles resolved at the sensor by the factor CNVRTCPM which, for most sensors, turns out to be the ratio of the narrow FOV to the wide FOV. If the value for CNVRTCPM is not specified, the model uses the value 1.0 so that the wide and narrow FOV are equivalent.

d. For a given sensor-target pair, the criterion ACQPERF for eventual detection has been stored as described in section 2.2.b. It is assumed that observers search using wide FOV and switch to narrow FOV to obtain higher levels of information about the target. A criterion for sufficient information to possibly engage a target (aimpoint) is currently set to be twice the value of ACQPERF. Targets whose cycles resolved in wide FOV meet the criterion for eventual detection and whose cycles resolved in narrow FOV meet the criterion for eventual aimpoint, are placed on the potential target list with an associated value which estimates their ease of detection. If the list is full (currently ten targets can be on a given observer's list), the least detectable target is removed to make room for one that is more detectable. Previously-detected targets are assumed to have priority over those not yet detected, and are always placed on the target list. The resolvable cycles presented by the target are modified by the probability of LOS, and the presence of large area smoke clouds is checked to determine their interference with detection. The attenuation along the LOS through large area smoke clouds is calculated in subroutines DOLASLOS and LASLOS. If the optical path length is above the threshold, LOS is assumed to be totally blocked. For lower values of the optical path length, the resolvable cycles presented to the target are calculated considering the degradations due to the smoke.

e. The computation of LOS (in subroutine DOLOS) is based on data stored in a grid of square cells, commonly 100 meters on a side. The elevation, the height of trees or urban buildings, and the code of the density of trees or buildings are stored. The obstruction height and the probability of LOS through the obstructions are provided by table look-up using the density code and the type of obstructions (trees or buildings). The locations

of the sensor and target are obtained as X, Y map coordinates and height above the ground. If the target location contains obstructions higher than the target, the probability of LOS (PLOS) is initialized to  $PL^2$ , where PL is the probability of LOS through the grid containing the target. Otherwise, PLOS is initialized to 1.0. (It is assumed that a target will use natural cover as much as possible, and the opportunity to use cover is assumed to be proportional to the same features affecting LOS. This is approximated by applying the factor PL twice to get the initial value.) The map coordinates are transformed to grid coordinates, and the difference in X, Y grid coordinates between the sensor and target locations is found. Each cell along the line from the sensor to the target is checked to see if the elevation blocks LOS. If not, the cell is checked for obstructions. If the obstructions in the subject cell are high enough to interfere with LOS, then the probability of LOS through those obstructions (PL) is multiplied by the old cumulative probability of LOS (PLOS). If the cumulative probability of LOS (PLOS) falls below 0.01, it is assumed that no LOS exists.

5. The calculation of the cycles resolved on a target is performed in subroutine PAIRS. Minimum and maximum range checks are done first. A parameter, currently set to 10 meters, defines the radius of certain detection. If the range is below this threshold, the number of resolvable cycles (BARS) is set to an arbitrarily large value. For ranges beyond detectability, BARS is set to zero. The contrast signature, whether thermal or optical, is attenuated as the energy is transmitted through the atmosphere and any obscurants which might be in the LOS. The effect of the attenuation can be expressed as

$$C_R = C_0 \cdot T_A(R) \cdot T_S \quad (18)$$

where  $C_R$  is the contrast at range R,  $C_0$  is the contrast at the target,  $T_A(R)$  is the transmittance through the atmosphere, and  $T_S$  is the transmittance through smoke clouds. The calculation in subroutine PAIRS is implemented in terms of the natural logarithms. For optical sensors,  $\log C_0$  is assumed to be the same for all targets, the extinction due to the atmosphere,  $\log T_A(R)$ , is obtained from the optical extinction curve initialized as in section 2.4, and the extinction due to large area smoke is provided by the subroutine DOLASLOS. The contrast provides the index to the sensor performance interpolation curve which was initialized as described in section 2.6, and the cycles per milliradian (CPM) are obtained from the linear interpolation as

$$CPM = SLOPE \cdot \log C_R + INTERCEPT \quad (19)$$

For thermal sensors,  $\log C_0$  is obtained from the contrast class of the target as in table 3,  $\log T_A(R)$  is just the extinction coefficient for the frequency band times range, and the attenuation due to large area smoke is provided as above. The CPM



are obtained from equation (19). For both optical and thermal sensors,

$$\text{BARS} = \text{CPM} \times \text{SIZE/RANGE}, \quad (20)$$

where SIZE is the minimum dimension in meters presented by the target, and range is in kilometers.

**3.4 Construction of Potential Target List.** If any targets pass the various filters, they are placed on the potential target list. If any of these targets has been detected in previous detection cycles, subroutine DOLASLOS is called to determine whether LOS has been broken by large area smoke clouds. If the previously detected target is still in LOS, the flag indicating detection is reset, otherwise the target is removed from the list of potential targets.

#### 4.0 DETECTION

Subroutine SEARCH is called twice every detection cycle, and SEARCH calls DODETECT to drive the detection process. DODETECT then calls subroutine DETECT for each side (for units which have possible targets) every other time called. Subroutine DETECT controls the detection process for each given unit and determines if a potential target should be detected now.

##### 4.1 Observer Unit Characteristics

a. Tests very similar to those described in section 3.1 and 3.2 determine whether or not the calculations for detection need be attempted. Destroyed units and those made inoperable by chemical weapons effects are skipped. Units which are mounted but can detect targets are given the appropriate characteristics of the host platform. The observer's location and sensor characteristics are determined. It is assumed that one-man units cannot search while firing or reloading.

b. The search pattern is assumed to be uniformly distributed throughout a gamer specified search area. The view-fan is specified by the gamer to have a direction with symmetric azimuthal limits. The sensor wide and narrow fields of view are input in radians by sensor type. If the wide FOV is very narrow (less than six degrees) the sensor FOV is reduced still further to partially account for the elevation part of the search. Currently, the basic search is implemented as almost totally azimuthal, so an elevation factor (sensor FOV/six degrees) is applied for very narrow FOV's to require longer times to search. The search by sensor FOV is implemented by assuming that the time available for search is uniformly distributed to the fraction of the total view-fan occupied by the sensor FOV. In other words, the probability the target is in the FOV is the sensor FOV divided by the search sector. The sensor FOV is also degraded by an input parameter if the chemical situation warrants. A moving unit,

except for flyers, searches for 180 degrees. This is different from the 360 degrees used in building the target list (section 3.2) and is merely a heuristic approach to reduce the time spent searching when the most threatening area is probably known. This search sector applies until the unit has been stopped for at least two minutes.

**4.2 Target Unit Characteristics.** For each observer that passes the filters, all the targets from the observer's potential target list are considered for detection. If any potential targets are destroyed or have mounted a vehicle, they are removed from the potential target list. Units which are already detected are not processed again. Line-of-sight is checked for the positions in the same way as described in section 3.3.c. The probability of LOS is used as a size factor in the calculation of the target's effective size. The target is further reduced to 1/3 its size if in hull defilade or 1/40 its size if in turret defilade. Moving targets are assumed to be easier to detect because of their increased signature, and their sizes are multiplied by 1.5 as discussed in section 3.3.b.(2). A target that has fired in the last twenty seconds is assumed to have cued the observer to search in its area, and the probability of being in the FOV of the sensor is set to 1.0. The attenuation due to large area smoke clouds is calculated for the existing LOS.

**4.3 Probability of Detection.** Subroutine PAIRS calculates the number of resolvable cycles (BARS) across the target transmitted to the sensors as described in section 3.3.d, then subroutine PDETSC is called to determine the probability of detection in the duration of this search cycle. The probability of detection given the target is in the FOV is given by equation (2) in section 2.3.

$$P(t) = 1 - \exp(-t/\text{TAU}), \quad (2)$$

The initialization of the time factor 1/TAU as a function of BARS is described in section 2.3, and the value is determined in subroutine PDETSC for the given BARS. The probability of equation (2) is then multiplied by the probability the target is in the FOV, to give the final probability of detection. A random draw is made to determine if the target may be detected by the sensor. If so, then smoke clouds are checked to see if LOS is blocked, and if it is, the target is removed from the potential target list. Otherwise the target is considered to be detected. Human targets are designated "recognized" when detected to represent the effect that foot soldiers can be distinguished from vehicles at this level of detection.<sup>9</sup>

#### 4.4 Impact of Detection on Other Functions

a. The detected unit will be engaged with direct fire if possible. The weapon selection is made for the firer-target environment by subroutine WTCWPN. If a weapon cannot be selected,

(out of ammunition, for example), no firing event is scheduled. If firing is in progress, the firing event is not scheduled for this detection. The time for the next direct-fire event is set in the queue, and modified for a chemical environment, if necessary.

b. After a unit has been detected, other criteria are checked to see if the resolution allows higher levels of inference to be made about the target. The number of cycles resolvable across the target by the sensor in narrow FOV are compared to the thresholds for this weapon target pair scaled first for recognition (3.5) and next for identification (6.4). This allows the graphical interface to present different icons to the gamers for the different levels of information perceived about the targets. Furthermore, the firing criterion can be changed by the gamer on Screen III.

## 5.0 SUMMARY OF ASSUMPTIONS

a. The body of heuristic methods and theory commonly known as the Center for Night Vision and Electro-Optics (CNVEO) search model represents the performance of sensors versus targets on the battlefield.

b. The fraction of observers who will eventually find a given target is given by

$$PINF = R \cdot E / (1 + R \cdot E)$$

where  $R = N/N50$  is the ratio of the number  $N$  of resolvable cycles across the target to the number  $N50$  of cycles required for PINF to be 0.5, and  $E = 2.7 + 0.7 \cdot (N/N50)$ .

c. The reciprocal of the mean time to detect ( $1/TAU$ ) is

$$1/TAU = PINF/3.4 \text{ for } N/N50 \leq 2,$$

and

$$1/TAU = N/(6.8 \cdot N50) \text{ for } N/N50 \geq 2.$$

d. The optical paths from target to sensor are near the horizon.

e. The detections occur in a homogeneous, natural atmosphere. (The sky is just as bright at range  $R$  from the target as at the target.)

f. The sky-to-ground ratio of luminances is the same for all sensor-target pairs.

g. The minimum contrast which can be detected by the eye is 0.02.

h. Targets within 10 meters are certain to be detected.

i. No LOS exists if the probability of LOS is less than 0.01.

j. The optical contrast at the target is the same for all targets.

k. The search pattern is uniformly distributed throughout the search area (view-fan).

l. A target that has fired in the last twenty seconds is assumed to have cued the observer to search in its area, and the probability of being in the FOV of the sensor is set to 1.0.

m. Moving observers are assumed to search 360 degrees while targets are being considered for the potential target list. This saves the gamers the effort of setting and adjusting view-fans while providing the approximate effect of having individual, coordinated search sectors for all vehicles in a moving organizational element.

n. Moving observers are assumed to search 180 degrees while targets are being considered for detection. This lessens the search time penalty assuming most units know the approximate direction of the threat.

## REFERENCES

1. W. R. Lawson, T. W. Cassidy, and J. A. Ratches, "A Search Prediction Model," Proceedings of the IRIS Specialty Group on Imaging, June 1978.
2. R. Flaherty and W. Lawson, "Laboratory Search and the Night Vision and Electro-Optics Laboratory Search Model," Proceedings of the IRIS Specialty Group on Imaging, January 1983.
3. John Johnson, "Analysis of Image Forming Systems," Proceedings of Image Intensifier Symposium, 1958.
4. S. R. Rotman and M. L. Kowalczyk, "Extending the CNVEO Search Model to the Multitarget Environment," Institute for Defense Analyses Paper P-2022, June, 1987.
5. J. A. Ratches, et.al., "Status of Sensor Performance Modeling at NV & EOL," Proceedings of the Infrared Information Symposium, NAVSO P-2315, Vol. 26, No. 1, June 1982.
6. Donald W. Hoock, "A Modeling Approach to Radiative Transfer Through Inhomogeneous Dust and Smoke Clouds," Proceedings of 7TH Annual E-O S&EL/TWT Conference, Vol. 2, R. Shirkey, editor, pp. 575-606, Feb 1987.
7. R. E. Turner, et.al., Model Development for E-O S&EL: Natural Aerosol Contrast, Laser Transmission and Turbulence, ASL-OR-80-0127-1, U. S. Army Atmospheric Sciences Laboratory, White Sands Missile Range, NM 88002, January 1980.
8. R.E. Huschke, Atmospheric Visual and Infrared Transmission Deducted from Surface Weather Observations: Weather and Warplanes VI, Rand Corporation Report R-2016-PR, Santa Monica, CA 90406, 1976.
9. B.L. O'Kane, M.D. Greshaw, J. D'Agostino, and D. Tomkinson, "Human Target Detection Using Thermal Systems," Proceedings of IRIS Passive Sensors, applied Physics Laboratory, Laurel, MD (Feb 92).

## APPENDIX B – SELECTED JANUS ALGORITHMS

Extracts from *Janus (T) Documentation and Users Manual*.  
Department of the Army, US Army TRADOC Analysis Command,  
White Sands Missile Range, NM; 1988.

### B-1 LINE OF SIGHT

#### B-1.1 GENERAL

Line of sight occupies a central role in combat simulations that work at item level. The line of sight algorithm is critical to both the processes being simulated, and to the overall ratio of real time to simulation time characteristics of the model. The overall purpose and use of the model, in turn, drive the selection of algorithms from the viewpoint of speed and accuracy. "Janus/T" leans more toward speed than very detailed calculations in most of the algorithms used. This section will discuss three aspects of LOS processing in "Janus/T". Those aspects are:

- LOS in support of detections
- LOS through smoke and dust clouds
- LOS support the force deployment

#### B-1.2 LOS FOR DETECTIONS

The principal problem to be solved is to determine if a line from an observer to a target intersects intervening terrain or terrain features. The process works as follows:

- In general, the line between the observer and target will be divided into equidistant points. Each point will be tested to determine the degree to which it influences LOS.
- The number and location of points tested along the observer target line (LOS Line) is determined as follows:

Compute observer to target  $[\Delta x]$  and  $[\Delta y]$

Compute  $N_x$  and  $N_y$  by dividing  $[\Delta x]$  and  $[\Delta y]$  by the terrain grid size

The largest  $N_x$  and  $N_y$  is called  $N_p$ .  $N_p$  is the number of points to be tested along the observer target line.

Compute  $d_x$  and  $d_y$  to be  $\frac{\Delta x}{N_p}$  and  $\frac{\Delta y}{N_p}$

The points to be tested along the observer target line are generated by successive additions of  $d_y$  and  $d_x$  to the observer's position.

Each point is tested in the following way: The height of the terrain at any point along the LOS line is the height of the terrain in the terrain grid cell containing the point. If the terrain is higher than the point on the observer target line, then the process is terminated and LOS is said not to exist between the observer and target.

Note that the observer target line is adjusted to account for the height of the observer and the height of the target.

If the observer target line does not intersect terrain, then it will be tested for being below any terrain features such as buildings or vegetation.

If the point on the observer target line is below terrain features, then the density of the terrain feature is added to a sum kept for the observer target line.

If this sum exceeds a threshold, then line of sight is said not to exist. Terrain feature density of the terrain grid cell containing the observer is not added to this sum. If the terrain feature densities do not exceed a threshold, they will affect the observer's ability to detect the target in the following way.

Let  $G$  be the grid size in Km and let  $D$  be the sum of the density codes of the cells for which LOS line is below the city or tree height.

The  $F = (G * D)^2$  and degradation =  $EXP(-F)$ . The table below contains examples of degradation computations for a grid size of fifty meters.

The degradation is applied to the target detection dimension during the acquisition process.

Density Code	Path Length in Grids	LOS Degradation
1	1	0.997503
1	2	0.990050
1	3	0.977751
1	4	0.960789
4	1	0.960789
4	2	0.852144
4	3	0.697676
4	4	0.527292
7	1	0.884706
7	2	0.612626
7	3	0.332040
7	4	0.140858

### B-1.3 LOS THROUGH SMOKE AND DUST

If LOS exists between an observer and target, the model also checks to see if smoke or dust block the LOS line. Assuming that there are currently some active clouds, the following process is used to determine if any could block line of sight.

- Each cloud is defined by the three dimensional location of its center, the orientation/direction of the length dimension and two sets of length, width, and thickness dimensions. One set of length, width, and thickness is for optical sensor, the other for thermal sensors. The set of dimensions used is determined by the sensor the observer is currently using. any intersection of the LOS line and the cloud blocks LOS.
- A gross filter is first applied to the cloud to see if it could block LOS. This is done by constructing square around the cloud. The dimension of the square is the largest of the length of width dimension of the cloud. Next, a rectangle is placed over the observer and target so that the observer and target lie on opposite ends of the rectangle's diagonal. If the cloud square does not intersect the observer target rectangle, then the cloud is no longer considered.
- Next, a finer filter is applied. The distance from the cloud center to the line connecting the observer and target is compared to the cloud's maximum dimension. If the distance is larger than the dimension, then the cloud is no longer considered.
- If the testing gets this far, a third test is made. This test uses the cloud's orientation, and length and width dimension to form a diamond. The four lines making up the diamond are tested for intersection with the LOS line. If no intersections occur, then the cloud is no longer considered.
- If the test above resulted in an intersection, then the altitude of the intersection is compared to the altitude of the top and bottom of the cloud. If it is between those two altitudes, then the cloud is declared to block LOS.

### B-1.4 LOS FOR DEPLOYMENT

The "LOS" routines can be used during the play of "Janus/T" to indicate the general line of sight characteristics for a unit located at a particular terrain location.

- To do this, the player picks "LOS" from menu section two. Next, the player picks a unit. Picking a unit establishes the view to be considered by the LOS routines. View is defined by view fan size, view fan direction, visibility limit and unit height. The view fan size, direction, and limit are drawn in white on the map display.

View fan size is the angle between the pie shaped wedge representing the view.

View direction is designated by the dashed line down the center of the view.



Visibility limit is represented by the circular arc at the far end of the view.

Unit height is not explicitly represented by the graphics.

Height and visibility limits are set by the system type of the unit.

View fan size and view fan direction can be set by the player.

- When the unit is picked (after picking LOS), the unit's view will be graphically displayed and the general LOS capability of the terrain location and unit picked will be indicated within the view by orange lines. The orange lines are generated using the following procedure. Starting from one side of the view, LOS as described above is tested every five degrees for points along a line from the view point to the view visibility limit. Whenever two consecutive points have line of sight, an orange line is drawn between the two points. Points between the five degree angle increments are not considered. The LOS degradation factor must be above 0.58 in order for the point to qualify as a point to which LOS exists.
- Once the unit has been picked and the LOS routines know the view definition, it is possible to test what this unit could see from other location. To do this, move the cursor to another location and press the yellow button. the line of sight capability of the unit from that point will then be displayed. In this fashion, the player can decide where to deploy each unit.

## B-2 ACQUISITION

The target acquisition algorithms used in "Janus/T" are based on the mathematical detection model developed by the Night Vision and Electro-Optics Laboratory, hereafter referred to as the NVEOL detection model. Actually, the NVEOL detection model is not a well-defined model, in that it is rather more like an approach, or concept, for modeling the detection/acquisition process. Documentation of the NVEOL detection model is scattered throughout various published reports, unpublished papers, letters, briefing slides, etc. This leaves the Force-on-Force Combat Simulation designer with considerable latitude in selecting and/or designing a specific implementation of the NVEOL detection model.

In the remainder of this section we attempt to describe, in sufficient detail, the implementation of the NVEOL detection model in "Janus/T". Modeling detection/acquisition in Janus/T" is divided into two basic parts: modeling the physical stimulus that a target presents to an observer, and modeling the observers response to that stimulus.

## B-2.1 STIMULUS PRESENTED BY TARGET

### B-2.1.1 Definition of Cycles *fine*

The physical effects that make a target visible are condensed into a single measurement. This measurement is the number of "cycles" resolved by the observer upon the target. Image a pattern of stripes or bars which are equal in width and alternating in color. Let the contrast between the two colors be the same as the contrast between the image of a target and its background. Make the overall size of the pattern the same as the target's minimum presented dimension. Let the spacing of the bars become finer and finer until we reach the finest spacing that the observer can distinguish. Count the number of bars in the pattern. This gives the number of cycles that the observer can resolve on the target.

### B-2.1.2 General method for Computing Cycles

The general method for computing cycles in "Janus/T" involves these steps:

- \* Find the contrast of the target as it appears to the observer's sensor.
- \* Find the number of cycles per milliradian that the sensor can resolve as a function of that contrast.
- \* Use the targets presented dimension and range to find the number of milliradians subtended by the target.
- \* Multiply cycles per milliradian times milliradians subtended by the target, to obtain cycles.

The next sections describe the mathematical model for computing cycles.

### B-2.1.3 Computing Contrast for Direct Vision Optical Sensor (DVOs)

The unaided eye and binoculars are examples of DVO sensors. We shall refer to such sensors as "optical sensors", and we will refer to targets under observation by these sensors as "optical targets". The procedure for determining the contrast of an optical target is as follows.

Let:

$CS$  = contrast of target at sensor

$CT$  = contrast of target measured at target

$SG$  = sky to ground brightness ratio in  $Km^{-1}$

$EX$  = atmospheric extinction for visible spectrum

$RA$  = sensor to target range in  $Km$

$$CS = CT / (1 + SG * (EXP(EX * RA) - 1)) \quad [\text{Eq. 1}]$$

The sources of the inputs to this calculation are:

- CT: This value is input as part of the weather data. (See table 28, the entry "OPTICAL CONTRAST") All optical targets are assumed to have the same value of CT.
- SG: This value is input as part of weather data. (See table 28) Although provision has been made to portray SG as a function of the observer-to-target direction, this has not yet been implemented in the "Janus/T" code. The value of SKY TO GROUND BRIGHTNESS for 90 degrees determines the value of SG used in "Janus/T".
- EX: This value is input as part of the weather data. (See Table 28, under the heading "EXTINCTION BAND 1")
- RA: This value is computed by the model from the current observer and target locations

#### B-2.1.4 Computing Contrast for other Sensors

Sensors such as the M-1 Tank thermal sight will be referred to as thermal sensors. These sensors operate on wavelengths roughly between 7 and 12 microns. Instead of the "contrast" of a target observed by a thermal sensor, we will speak of the absolute value of the average target to background temperature difference or "delta-T" of the target.

Let:

ST = absolute avg. target to back ground temp difference measured at sensor, deg C

TT = absolute avg. target to background temp difference measured at target, deg C

EX = atmospheric extinction for 7-12 micronband, in units of inverse Km

RA = sensor to target range, Km

Then:

$$ST = TT * EXP(-EX * RA) \quad [\text{Eq. 2}]$$

The sources of the inputs to this calculation are:

- TT = This depends on the "contrast class" which has been assigned to the target system. (See Table 13, the entries "EXPOSED THERMAL CONTRAST CLASS" and "DEFILADE THERMAL CONTRAST CLASS")

The translation between contrast class and deg C is:

Class

Deg C

1	0.5
2	1.0
3	1.5
4	2.0
5	2.5
6	3.0
7	3.5
8	4.0
9	4.5
10	5.0
11	6.0
12	7.0

EX: This input is from the weather data. (See Table 28, the entry "EXTINCTION BAND 2") Although the 7 to 12 micron band is called "BAND 2" in "Janus/T", this band is usually called band 5 in other programs, such as Atmospheric Science Lab's EOSAEL model.

RA: This is computed by the model from the sensor and target locations.

#### B-2.1.5 Computing Cycles Per Milliradian

The performance of a sensor is specified by a curve of cycles per milliradian as a function of target contrast measured at the sensor (in the case of DVOs), or as a function of the absolute average target to background temperature difference in degrees C (in the case of thermal sensors). This data is entered in Table 31. The columns marked "CYCLES" refer to cycles per milliradian data. Those marked MRT?MRC refer to contrast or delta-T data. (MRT abbreviates "mean resolvable temperature"; MRC abbreviates "mean resolvable contrast". Sensor performance is therefore specified by an MRT or an MRC curve.

Before entering these curves, the user should determine whether the magnification of the sensor is represented by the data he intends to use. A Janus sensor is not simply a type of sensor such as a dragon thermal sight, but it is a particular type of sensor operating in a particular field of view. Magnification depends on which field of view is assumed. Because of the way acquisition is modeled (see "CHOICE OF SENSOR FIELD OF VIEW" below) it is best to assume sensors operate in their narrowest field of view. In past studies at TRASANA, the MRT curves for thermal sensors have included this magnification. Some of the DVO curves have included it, while others required that the user adjust a curve for the eye by multiplying cycles per milliradian data by sensor magnification. The "Janus/T" code does not perform

such adjustments; these must be done by the user before the sensor performance data is entered.

"Janus/T" assumes the minimum usable contrast for a DVO is 0.02. No sensor performance curves for DVOs should be used with MRC values less than 0.02. Failure to observe this may cause "Janus/T" to bomb during initialization.

#### (6) Computing Minimum Presented Dimension of Target

The minimum presented dimension of a target is computed as a function of the detection dimension of the target, the terrain which lies along the sensor-to-target line of sight (LOS), and whether the target is in defilade or not.

Let:

AD = actual minimum presented dimension of target, meters

DD = detection dimension of target, meters. This should be the minimum presented dimension of the target when it is completely exposed.

F1 = factor depending on whether target is in defilade or not.

F2 = factor depending on terrain along LOS.

Then:

$$DA = DD * F1 * F2 \quad [\text{Eq. 3}]$$

The sources of these inputs are:

DD: This is data entered for each system. (See Table 13, the entry "MINIMUM DIMENSION". Do not confuse this data item with "HEIGHT".)

F1: F1 is set to  $\frac{1}{3}$  if target is in defilade, set to 1.0 otherwise

F2: This factor is computed by the LOS algorithm. (See the preceding section "PROBABILITY OF LOS")

#### B-2.1.6 Computing Milliradian Subtended by Target

Let:

MR = milliradian subtended by target

AD = actual target dimension in meters

RA = sensor to target range in kilometers

Then:

$$MR = AD / RA \quad [\text{Eq. 4}]$$

The input AD is described above, RA is computed from the sensor and target locations. ( $AD/RA$  is an approximation for MR.)

As stated earlier, no adjustment is made to AD or MR to account for sensor magnification. "Janus/T" assumes such an adjustment is part of the MRT or MRC data for the sensor.

#### **B-2.1.7 Algorithms Used by "Janus/T"**

It is more convenient to work in terms of the logarithms of many of the quantities mentioned above than in terms of the quantities themselves. During "Janus/T" initialization, much of the above data is converted by taking logarithms. For example, the sensor performance curves are used to generate curves which represent cycles per milliradian as a function of the natural logarithm of contrast or temperature.

The curve corresponding to Eq. 1 is generated as a set of slope-intercept interpolation data during initialization.

#### **B-2.2 OBSERVERS RESPONSE TO TARGET STIMULUS**

Each 20 seconds of Game Time, a list of at most 5 potential target units is formed for each observer unit. We will now describe the requirements which must be satisfied in order for an enemy unit to be included in this list. The observer must have LOS to the target, and must resolve a sufficient "cycle ratio" on the target unit. The target unit must be in the observer's sector of search. If there are more than 5 targets meeting these conditions, only the 5 with the most cycles are put in the list. Cycle ratios and search sectors are described in more detail in later sections.

A unit in "Janus/T" may have two sensors. An observer unit uses only one of these sensors at a time. If no targets are currently acquired by an observer, "Janus/T" will cause the unit to switch to its other sensor before its potential target list is updated.

Each 2 seconds of Game Time, an observer unit attempts to acquire those units in its list of potential targets. An independent  $U(0,1)$  random draw is made for each potential target, and compared to a calculated probability of detection for that target. (Details of computing the probability of detection are given in a later section.) If the random draw is less than or equal to the computed probability of detection, the target unit is acquired, otherwise it is not. Once acquired, the target unit will remain acquired until either LOS is lost, or the target is no longer within the observer unit's search sector.

### B-2.2.1 Threshold Cycle Ratio

During "Janus/T" initialization, each possible combination of observer unit with enemy target unit is assigned a "threshold cycle ratio" by a random drawing. The distribution of threshold cycle ratios is given as follows:

Let:

$PR$  = probability that assigned threshold cycle ratio is less than or equal to  $R$

$TR$  = value of threshold cycle ratio

$A = 2.7$

$B = 0.7$

$W = A + B * TR$

$TERM = (TR)^2 W$

Then:

$$PR = TERM / (1 + TERM) \quad [\text{Eq. 5}]$$

This distribution of threshold cycle ratios corresponds to the "P=INFINITY" probability distribution in the VNEOL detection model.

As was stated above, one of the requirements for an enemy unit to be included in an observer unit's potential target list, is that the observer must resolve a sufficient "cycle ratio" on the target unit. This requirement is evaluated in the following manner. Suppose the observer resolves  $C$  cycles upon the target. The value of the "cycle ratio" is then  $CR = C / M$ , where  $M = 3.5$  if the target is stationary, or  $M = 2.0$  if the target is moving or has fired in the previous 15 seconds. In the NVEOL Model,  $M$  is called the "cycle criterion". It is the median number of cycles required for eventual acquisition. Using  $M = 3.5$  corresponds to acquisition at the level of "recognition", which means the observer acquires the target and knows whether it is tracked or wheeled or personnel. Using  $M = 2.0$  corresponds to acquisition at the level of "aimpoint", which means the target is acquired and can be aimed at.

The computed "cycle ratio" (CR) is then compared to the threshold cycle ratio which was assigned to the observer/target combination during "Janus/T" initialization. The computed "cycle ratio" must be greater than or equal to the threshold cycle ratio, before the target is eligible to be put in the observer's potential target list.

(The above mathematical model is actually implemented in "Janus/T" by assigning threshold values of  $TR * 3.5$  to  $TR$ . Then if a target has moved or fired,  $C$  is multiplied by 1.75 and compared to the threshold. This is equivalent to comparing  $C / 2.0$  to  $TR$ .)

We emphasize that a threshold cycle ratio is assigned to each observer-unit/target-unit combination during initialization. Once set, these thresholds are not changed during a particular "Janus/T" run.

#### B-2.2.2 Search Sector

If an observer unit is not in defilade, its search sector is a circle centered at the unit, with a radius of "VISIBILITY". The data item "VISIBILITY" is described in Table 13.

If an observer unit is in defilade, its search sector is a pie shaped wedge, with the point of the wedge located at the unit's position. The wedge is called the unit's "View Fan". The angular extent and direction of the unit's view fan is set by the player, and the radial extent of the fan is determined by the "VISIBILITY" of the observer unit.

A moving unit will not restrict its search to the View Fan set by the player, since such a unit is not in defilade. Also, when a unit stops moving, there may be a time delay before it goes into defilade and its View Fan becomes effective.

#### B-2.2.3 Probability of Acquiring a Potential Target

Let:

$PD$  = probability target is acquired

$T = 2.0 \text{ sec}$

$M = 0.5$  cycles if target is moving or has recently fired,  $M = 2.0$  cycles otherwise

$C$  = cycles present on target

$CR = C / M$

$PF$  = a function of  $TR$  given below

$SS$  = angular extent of sector of search

$FV$  = field of view of observers sensor

Then:

$$P = 1.0 - \text{EXP}(-(FV / SS) * (PF / 3.4) * T) \quad [\text{Eq. 6}]$$

The value of  $SS$  is the angular extent of the observer unit's View Fan if in defilade, and 180.0 degrees is not in defilade. Note that "Janus/T" does not explicitly model the direction of the observer's sensor as a function of time. The factor  $FV/SS$  is the fractional amount of time that a particular target is in the observer unit's sensor field of view. (The value of  $FV$  is entered in Table 30.)



The values used for  $M$  correspond to the cycle criterion for acquiring a target at the level of "detection", which means that target is known to be a point of military interest. These values are appropriate for medium clutter conditions.

Now we describe the function  $PF$ .

Let:

$C$  = cycles resolved on target

$M = 2.0$

$CR = C / M$

$A = 2.7$

$B = 0.7$

$W = A + B * CR$

$TERM = (CR)^2 W$

Then if  $CR$  is less than or equal to 2:

$$PR = TERM / (1.0 + TERM) \quad [Eq. 7]$$

Else:

$$PF = CR / 2 \quad [Eq. 8]$$

### B-2.3 ACQUISITION BY FLYERS

When a flyer unit is not currently "popped up", its acquisition of enemy units is modeled in the same fashion as non-flyer units with the exception that the search sector of a moving flyer is set to 180 degrees instead of 360 degrees.

When a flyer unit is popped up, its acquisition process is modified, as follows.

#### B-2.3.1 Non-Scanning Flyers When Popped-Up

When a flyer that does not have an automatic target scanner is popped up, its list of potential targets is updated immediately, instead of waiting for the next normal 20-second update. Acquisition of enemy target units then proceeds in the same manner as for non-flyers.

#### B-2.3.2 Scanning Flyers When Popped-Up

When a flyer with an automatic target scanner is popped up by the player, it remains up for a certain time which simulates using an automatic scanner to capture a picture of the scene. The observer unit's list of potential targets is immediately updated, based on the performance of the narrow field of view of a special sensor which simulates the automatic scanner capabilities.

The flyer then pops-down (remasks) automatically. "Janus/T" simulates the pilot's search of the stored picture. The pilot considered to be using the wide field of view of the special scanning sensor used to build the potential target list, and is considered to be searching for a time period which is proportional to the angular extent of the View Fan of the flyer (which has been set by the player). An independent random draw is made to determine whether the pilot detects each target. The probability of detection is given by Eq. 6, with the following modifications made to the inputs of Eq. 6. The value of M used is 3.5 unless the target has fired. This accounts for the fact that movement is not evident on the stored picture. The value of T is one-half the actual time spent searching. This accounts for the fact that the pilot's search process is believed to be slower than a search of an actual scene.

The flyer will remain popped down for a time equal to the pilot's actual search time, plus 2 seconds for each target acquired. The additional time accounts for spending two seconds per acquired target to switch to narrow field of view and recognize the target.

If the pilot acquired no targets during the remark time period, the scanning procedure is started over; that is, the flyer pops up again and captures another picture of the scene, then remasks and the pilot again searches the stored picture for targets, etc.

If the pilot acquired targets during the remark time period, then the flyer will pop up and attempt to re-acquire all targets previously acquired from the scanner picture. The sensor the pilot uses will (be) the first sensor assigned to the system. (See Table 22, the entry 'FIRST SENSOR TYPE'.) Each 2 seconds, an independent random draw is made for each potential target. The probability of detection is given by Eq. 6. However, the ratio (FV/SS) in equation 6 is set equal to 1.0, since the pilot knows the location of the potential targets and can immediately place them within his sensor's FOV.

#### **B-2.4 EFFECT OF AGGREGATION UPON ACQUISITION**

A unit which is an aggregate of several systems acquires targets as if it were a single system. Such a unit is also acquired as if it were a single system. That is, when it is viewed as a target, it appears to be a single system with the dimensions and contract of a single system.

#### **B-2.5 CHOICE OF SENSOR FIELD OF VIEW**

The assumed FOV for a sensor is determined by what data the user decides to enter in Table 31. Since the level of acquisition required for putting a target in the observer's potential target list is sometimes as high as "recognition", sensors should

operate in their narrowest field of view in order to produce as many cycles as possible on the target. Once a target is in the list, the time needed to detect it depends on Eq. 6. Although using a narrow FOV decreases the term (FV/SS) in Eq. 6, it also increases the value of CR and PF, which somewhat compensates. By using Eq. 6 we are assuming that the elevation angle of the sensors field of view is always great enough to cover the elevation of the observer's search sector. If this were not the case, then the term (FV/SS) would have been replaced by the ratio of the area (in  $\text{deg}^2$ ) of the sensors FOV to the area of the search sector.

## **B-3 DIRECT FIRE**

### **B-3.1 GENERAL**

The Direct Fire Module (DFM) of "Janus/T" is invoked to process a single potential direct fire event, up to and including the outcome of the firing event itself. The DFM is initially scheduled (for a particular firing unit) by the Detection Module whenever a direct fire unit detects an enemy unit, but may subsequently re-schedule itself (for the same firing unit) if required.

In general terms, the DFM performs the following tasks each time it is invoked for a specified firing unit:

- Determine if the firing unit can fire at this time.
- Perform target and weapons selection, if possible.
- Fire one round (burst) and record the firing event.
- Decrement the appropriate ammo count for the firing event.
- Calculate the round (or burst) impact time.
- Make a preliminary determination of the number of kills. At impact time, determine if the kills are still valid. Record the kills, if any.
- Calculate the earliest next possible firing time for this firing unit.
- Re-schedule the DFM to run again for this same firing unit, if appropriate.

A detailed functional description of each of the above DFM tasks is presented in the following paragraphs.

### **B-3.2 INITIAL FIRING REQUIREMENTS**

Before being processed through the remainder of the DFM, a unit must satisfy all of the following requirements:

- The unit must be alive and operable
- The unit must be Direct Fire capable

## APPENDIX L

This Appendix provides a brief description of the content and usefulness of the publications listed in the bibliography.

Baer, W., and Akin, J.R., "An approach for real-time database creation from aerial imagery," Nascent Systems Development Inc., Carmel Valley, California.

This article discusses the real time generation of a of a perspective view database. The method of visually obtaining the terrain data is looked at as well as the construction of the Pegasus database. The article is geared primarily towards describing how a perspective view system could be generated.

Celski, R.J., "A Study of the Line of Sight Calculations and Data Base for the Janus(A) Model," paper for TRAC Monterey, Monterey, CA, 16 September 1992.

A report conducted to analyze problems in the representation of vegetation as well as its density and height values. The report looks at the HITREE and DOLOS subroutines and examines their flaws.

Digital Equipment Corporation, Order Number:AA-D034E-TE, VAX FORTRAN: *Language Reference Manual*, Maynard, Massachusetts, June 1988

Technical manual for understanding the VMS VAX FORTRAN language.

Kellner, A.D., "NVEOD Detection in Janus(A)," memorandum for the record for U.S. Army TRADOC Analysis Command, White Sands Missile Range, New Mexico, 14 July 1992.

Provides a summary of how detection and acquisition are calculated in Janus. Study of this memorandum will provide the reader with the necessary background knowledge required to understand the logic behind detection and acquisition in Janus. This memorandum is clearly written to educate the layman. It provides the basis of the background information on Janus in Chapter I of this thesis.

Macchiaroli, C.R., *A Review of Literature on the Theory of Visual Target Detection Probabilities*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1973.

As the title states, this thesis conducts a review of the literature available on target detection. This may provide a starting point for further understanding of the various ideas and approaches on target detection. However, this thesis was written over 20 years ago, and does not cover recent literature.

Parish, R.M., and Kellner, A.D., "Target Acquisition in Janus Army," paper for the U.S. Army TRADOC Analysis Command, White Sands Missile Range, New Mexico, October 1992.

This article covers the same topics as does Kellner's memorandum of 14 July mentioned above. It also describes some of the mechanics in Janus such as search initialization and generation of potential target lists. It is not an exact duplicate of Kellner's memorandum, and thus may offer additional insight into detection and acquisition.

TITAN Tactical Applications, *Software Design Manual Janus(A) 2.1 Model*, Contract No. DABFT 60-90-D-0002, Delivery Order #37.

Provides a description of the overall Janus system. Some of the more useful items included:

- description of data files and what they contain
- description of the Global files contained in Janus
- description of Janus algorithms. This includes discussions of radars, barriers, chemical warfare, smoke clouds, mine fields, movement, and line of sight.

Also included is a section based on the paper done by Kellner and Randall mentioned above.

TITAN Tactical Applications, *User Manual Janus 3.0 Model*, Contract No. DABFT 60-90-D-0002, Delivery Order #37.

Describes how to plan, execute, verify, and post process Janus scenarios. A useful manual if you want to create your own scenario.

TITAN Tactical Applications, Software Programmers Manual Janus (A)  
2.1 Model, Contract No. DABFT 60-90-D-0002, Delivery Order #37.

Provides detailed descriptions of all subroutines used in the Janus model. Also provides block diagrams showing interrelationships between subroutines. In each subroutine description is included a list of all subroutines and function that it calls and a list of any subroutines it is called by. An excellent tool for determining relationships within Janus.

*Janus (T) Documentation and Users Manual*, Department of the Army, US Army TRADOC Analysis Command, White Sands, New Mexico, 1988.

Describes another method of determining the line of sight and detection. The Janus (T) algorithm is designed more for speed than detail. Useful to read to get another perspective on how attenuation is calculated. The manner in which Janus (T) calculates attenuation seems more correct than that used in Janus (A).

#### Janus Programs

The programs themselves are full of comments and descriptions which are very useful in understanding what the particular routine is trying to do. This is a very useful tool when combined with the Software Programmers manual

## BIBLIOGRAPHY

Baer, W., and Akin, J.R., "An approach for real-time database creation from aerial imagery," Nascent Systems Development Inc., Carmel Valley, California.

Celski, R.J., "A Study of the Line of Sight Calculations and Data Base for the Janus(A) Model," paper for TRAC Monterey, Monterey, California, 16 September 1992.

Digital Equipment Corporation, Order Number:AA-D034E-TE, VAX FORTRAN: *Language Reference Manual*, Maynard, Massachusetts, June 1988

Kellner, A.D., "NVEOD Detection in Janus(A)," memorandum for the record for U.S. Army TRADOC Analysis Command, White Sands Missile Range, New Mexico, 14 July 1992.

Macchiaroli, C.R., *A Review of Literature on the Theory of Visual Target Detection Probabilities*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1973.

Parish, R.M., and Kellner, A.D., "Target Acquisition in Janus Army," paper for the U.S. Army TRADOC Analysis Command, White Sands Missile Range, New Mexico, October 1992.

TITAN Tactical Applications, *Software Design Manual Janus(A) 2.1 Model*, Contract No. DABFT 60-90-D-0002, Delivery Order #37.

TITAN Tactical Applications, *Software Programmers Manual Janus(A) 2.1 Model*, Contract No. DABFT 60-90-D-0002, Delivery Order #37.

TITAN Tactical Applications, *User Manual Janus 3.0 Model*, Contract No. DABFT 60-90-D-0002, Delivery Order #37.

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2  
Cameron Station  
Alexandria, Virginia 22304-6145
2. Library, Code 52 2  
Naval Postgraduate School  
Monterey, California 93943-5002
3. Department Chairman, Code ME 1  
Department of Mechanical Engineering  
Naval Postgraduate School  
Monterey, California 93943-5000
4. Professor Morris R. Driels 2  
Code ME/Dr  
Department of Mechanical Engineering  
Naval Postgraduate School  
Monterey, California 93943-5000
5. Frederick W. Dau IV 2  
101 Old Ferry Road #30D  
Shalimar, Florida 32579
6. Naval Engineering Curricular Officer, Code 34 1  
Department of Mechanical Engineering  
Naval Postgraduate School  
Monterey, California 93943-5004
7. Director, TRAC-MTRY 2  
Box 8692  
Naval Postgraduate School  
Monterey, California 93943
8. Adj. Professor Judith Lind 1  
Code OR/Li  
Department of Operations Research  
Naval Postgraduate School  
Monterey, California 93943-5000
9. Adj. Professor Bard Mansager 1  
Code MA/Ma  
Department of Mathematics  
Naval Postgraduate School  
Monterey, California 93943-5000
10. Adj. Professor Wolfgang Baer 1  
Code CS/Ba  
Department of Computer Science  
Naval Postgraduate School  
Monterey, California 93943-5000





DUDLEY KNOX LIBRARY  
NAVAL POSTGRADUATE SCHOOL  
MONTEREY CA 93943-5101

GAYLORD 5

DUDLEY KNOX LIBRARY



3 2768 00038476 2